Adonyi Róbert, Bertók Botond, Friedler Ferenc, Heckl István,
Hegyháti Máté, Holczinger Tibor, Imreh Csanád,
Kovács Zoltán, Süle Zoltán

Modeling and Optimization of Engineering Systems

A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja

TÁMOP-4.1.2.A/1-11/1-2011-0104

Főkedvezményezett:
Pannon Egyetem
8200 Veszprém
Egyetem u. 10.

Kedvezményezett:
Szegedi Tudományegyetem
6720 Szeged
Dugonics tér 13.

2014

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

COMBINATORIAL FRAMEWORK FOR PROCESS DESIGN AND SYNTHESIS

OUTLINE

- GENERAL INTRODUCTION
- ILLUSTRATION OF THE THEORETICAL AND PRACTICAL DIFFICULTIES IN ALGORITHMIC PROCESS SYNTHESIS
- COMBINATORIAL TECHNIQUE IN PROCESS DESIGN AND SYNTHESIS
- ADDITIONAL APPLICATIONS
- SCHEDULING OF MULTIPURPOSE BATCH PLANTS
- CONCLUDING REMARK
- REFERENCES

GENERAL INTRODUCTION

COMBINATORIAL PROBLEMS IN CHEMICAL ENGINEERING

- Almost all problems in chemical engineering especially in process design and operations, involve major (or at least some) combinatorial aspects.
- Examples
  - Process synthesis
  - Reaction-pathway (mechanism) identification
  - Scheduling
  - Molecular design
    etc.

CONVENTIONAL APPROACH TO SOLVE PROCESS DESIGN AND OPERATIONS PROBLEMS

- Formulation as a general mathematical programming problem (e.g., MILP, MINLP, NLP)
- Application of a general-purpose solver (e.g., GAMS)

- Outcome:

  A practical problem is too complex for the solver
  or
  a solvable problem is too simple to be practical.

CONVENTIONAL APPROACH TO SOLVE PROCESS DESIGN AND OPERATIONS PROBLEMS (Cont'd)

- Solution 1
  Awaiting for a faster computer.
  It may be futile: A computer thousand times faster can solve a problem with only 10 additional binary variables.

- Solution 2
  Compromising on the quality of the mathematical model.

- Solution 3
  Exploit the specific structure of the problem to accelerate the search.

OUR GENERAL PHILOSOPHY IN SOLVING COMPLEX PROCESS DESIGN AND OPERATIONS PROBLEMS

To develop

a problem formulation that manifests the unique structure of the class of problems

and

a solution procedure that exploits the specific structure of the problem.

- Result:
  enormous acceleration (industrial problems become solvable)
- Requirement:
  in-depth understanding of both the engineering and mathematical aspects of the problem
- The efficacy of our paradigm will be illustrated with process synthesis and scheduling.

ALGORITHMIC PROCESS SYNTHESIS

Given:

set of products,
set of raw materials,
mathematical models of the operating units

Generate:

optimal process or
best processes or
every feasible process

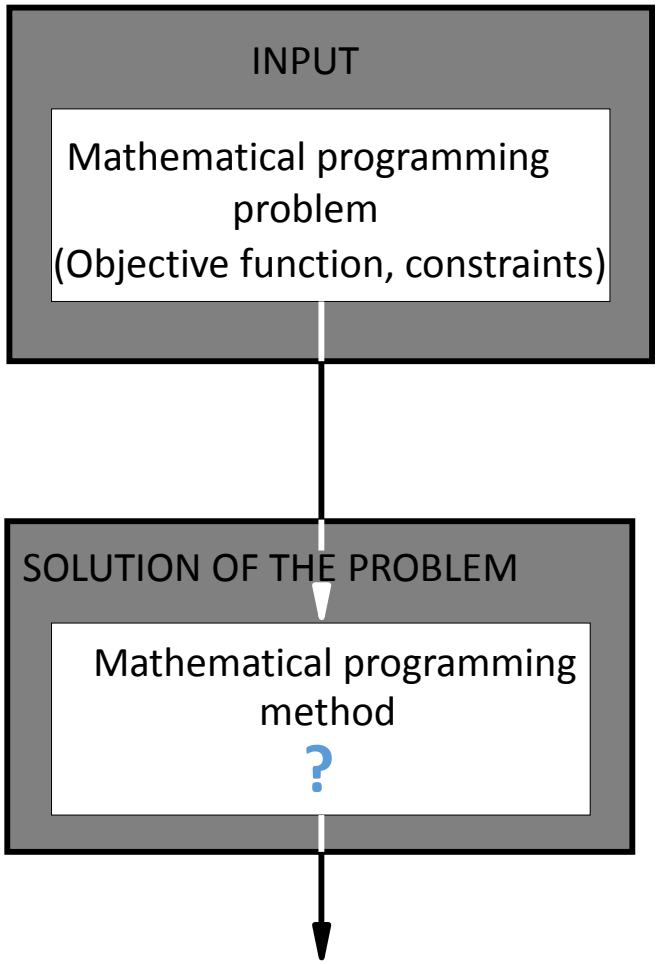
Optimality criteria:

cost, waste generation, controllability, risk,
combinations of them

QUESTION

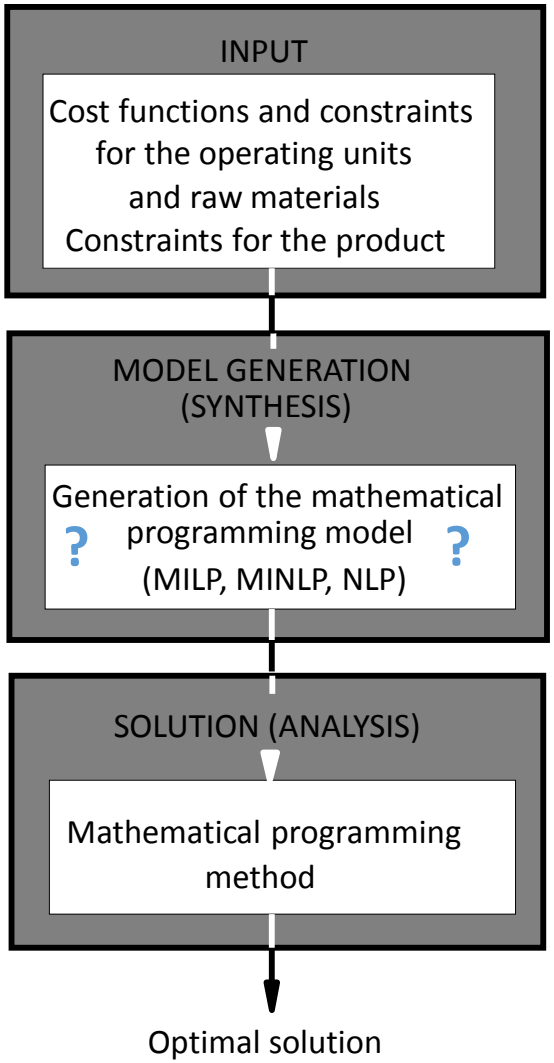
Is there any method for algorithmic process synthesis?

CONVENTIONAL MATHEMATICAL PROGRAMMING PROBLEM



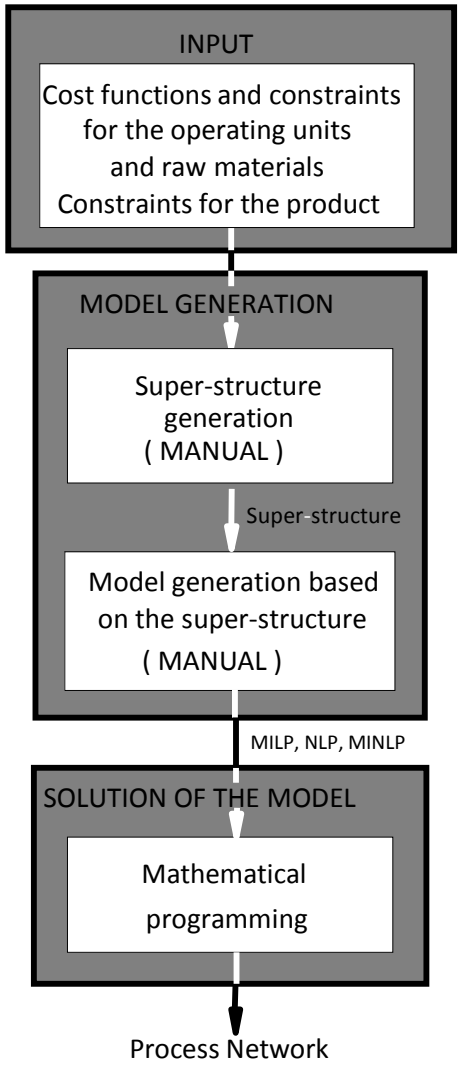
Comment: It is unsuitable for process synthesis.

ALGORITHMIC PROCESS SYNTHESIS



Comment: Model generation is the heart of a synthesis problem.

ALGORITHMIC PROCESS SYNTHESIS



Comment: The major activity is performed manually.

RIGOROUS SUPER-STRUCTURE

Super-structure that guarantees the optimality.

FORMAL DEFINITION: RIGOROUS SUPER-STRUCTURE

- Process synthesis problems are not specified as standard optimization problem (objective function and constraints).
- Suppose that systematic procedure is available so that a valid mathematical programming model can be generated for a network of the given operating units.
- A network of operating units is defined to be a rigorous super-structure if the optimality of the resultant solution cannot be improved for any instance of the class of problems by any other procedure for network and model generation.

HOW CAN A RIGOROUS SUPER-STRUCTURE BE GENERATED

- Step1. Exploring key features of the structures of feasible processes (or the structures of optimal processes) which are valid for each instance of the class of problems.
- Step2. Developing an algorithm that can generate a network including all structures possessing every key feature of these structures.

- Illustration: separation network synthesis

DIFFICULTIES IN ALGORITHMIC PROCESS SYNTHESIS: ILLUSTRATION BY SEPARATION NETWORK SYNTHESIS

PROBLEM DEFINITION

Given:

multicomponent feed-streams,

single or multicomponent product-streams,

operating units (separators, dividers, mixers)

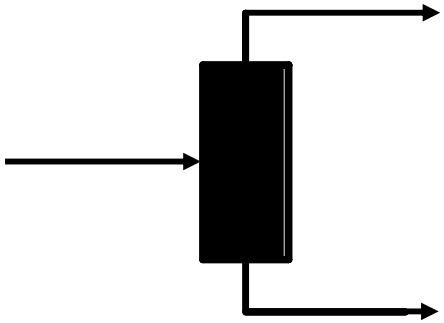
Generate:

the cost-optimal network

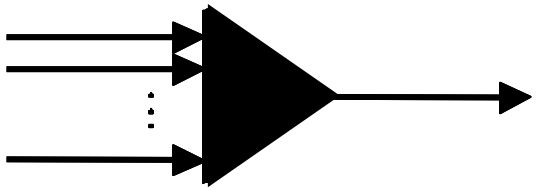
OPERATING UNITS

Separator

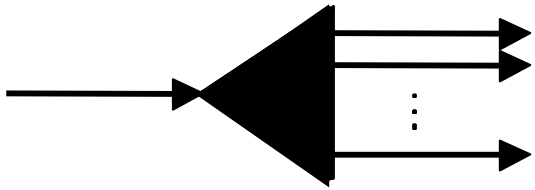
simple and sharp



Mixer

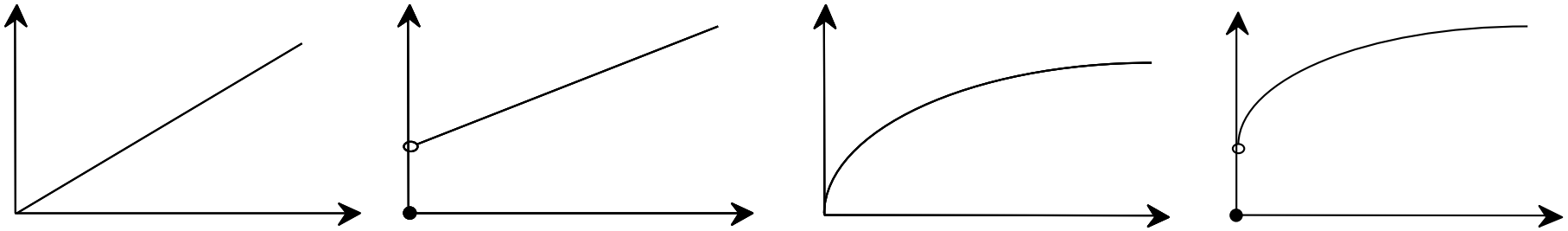


Divider



COST FUNCTION OF A SEPARATOR

- Concave
- Strictly monotone increasing
- Zero for zero mass-load



SEPARATION NETWORK SYNTHESIS (SNS)

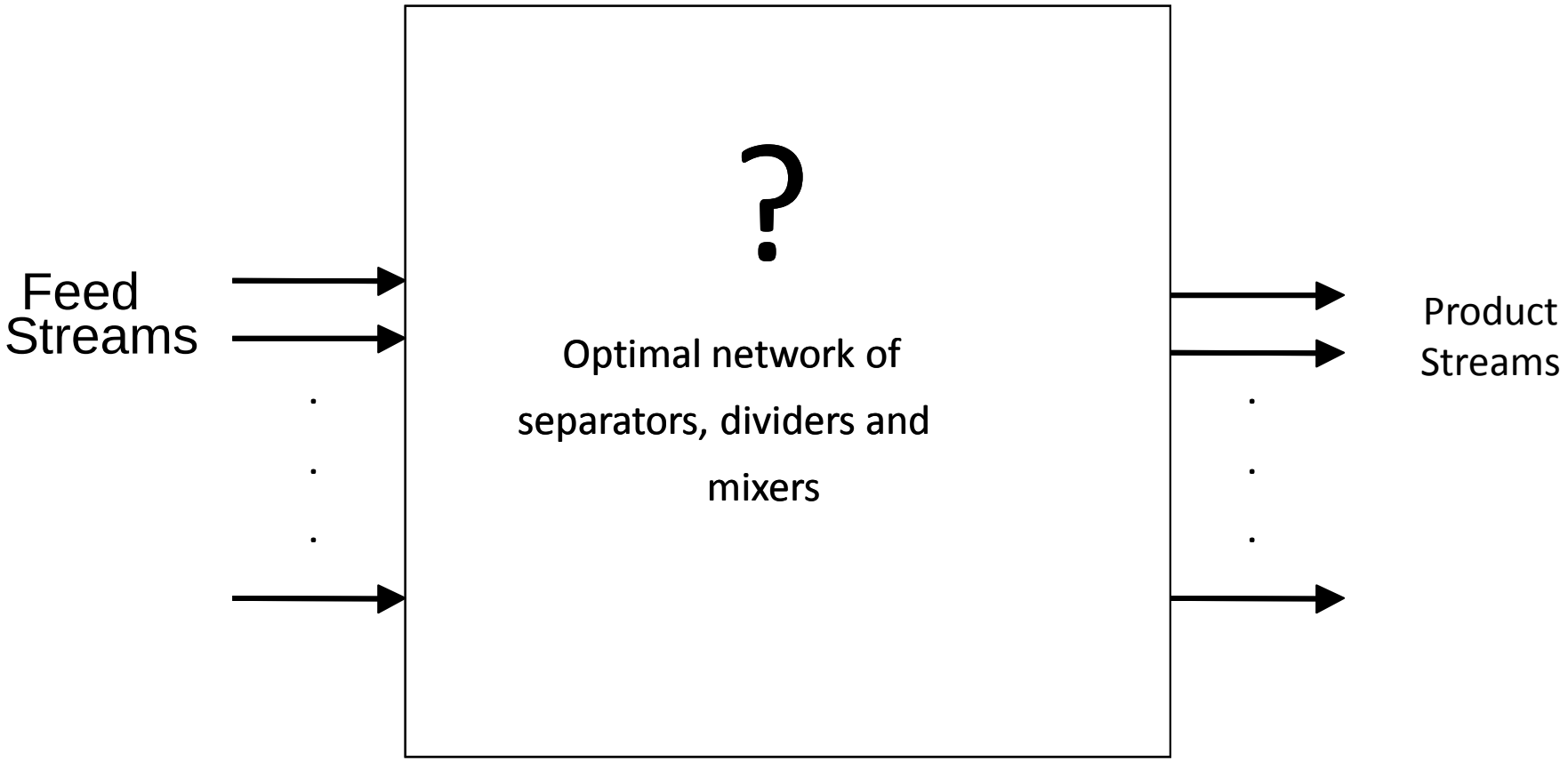
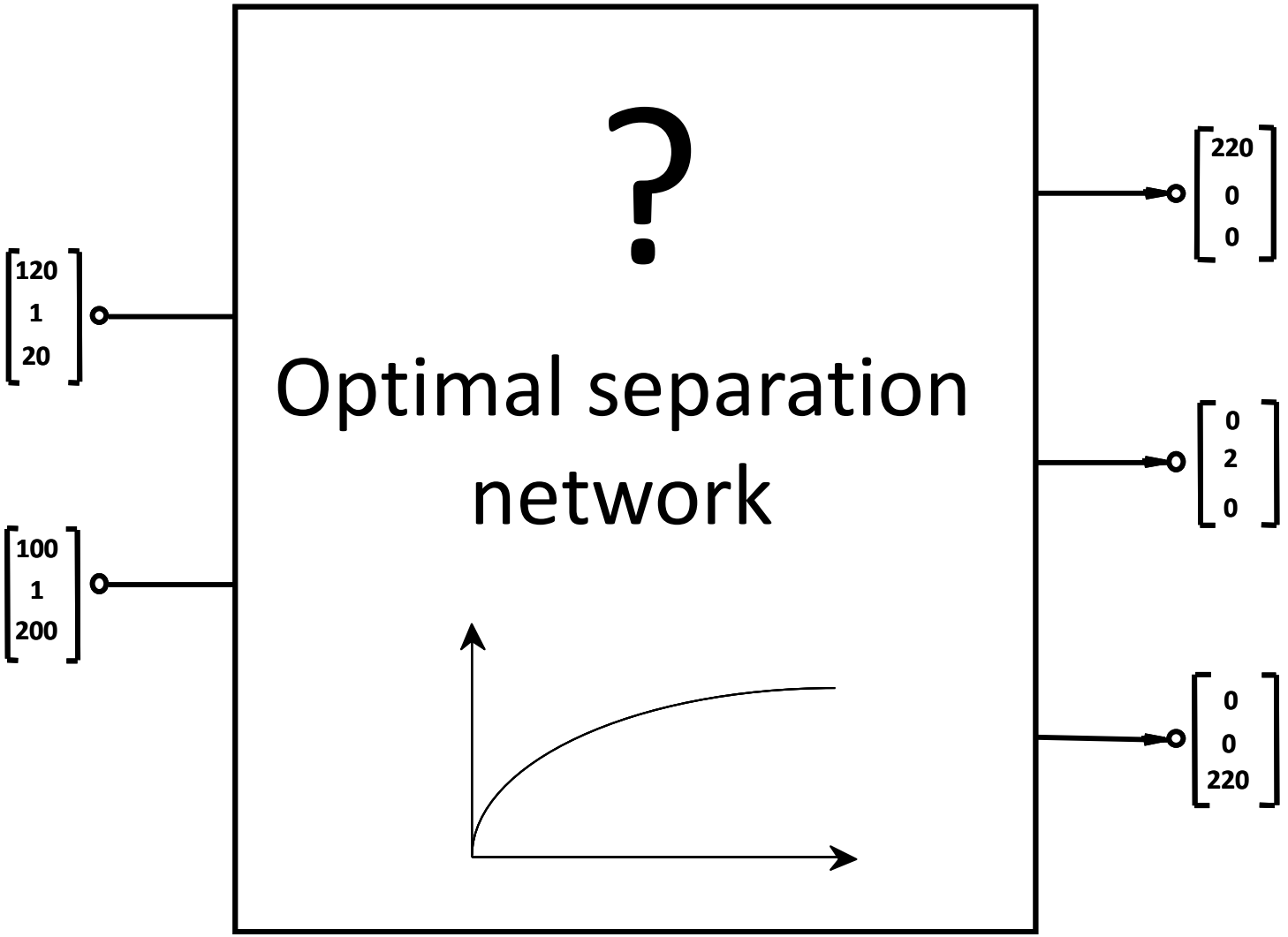
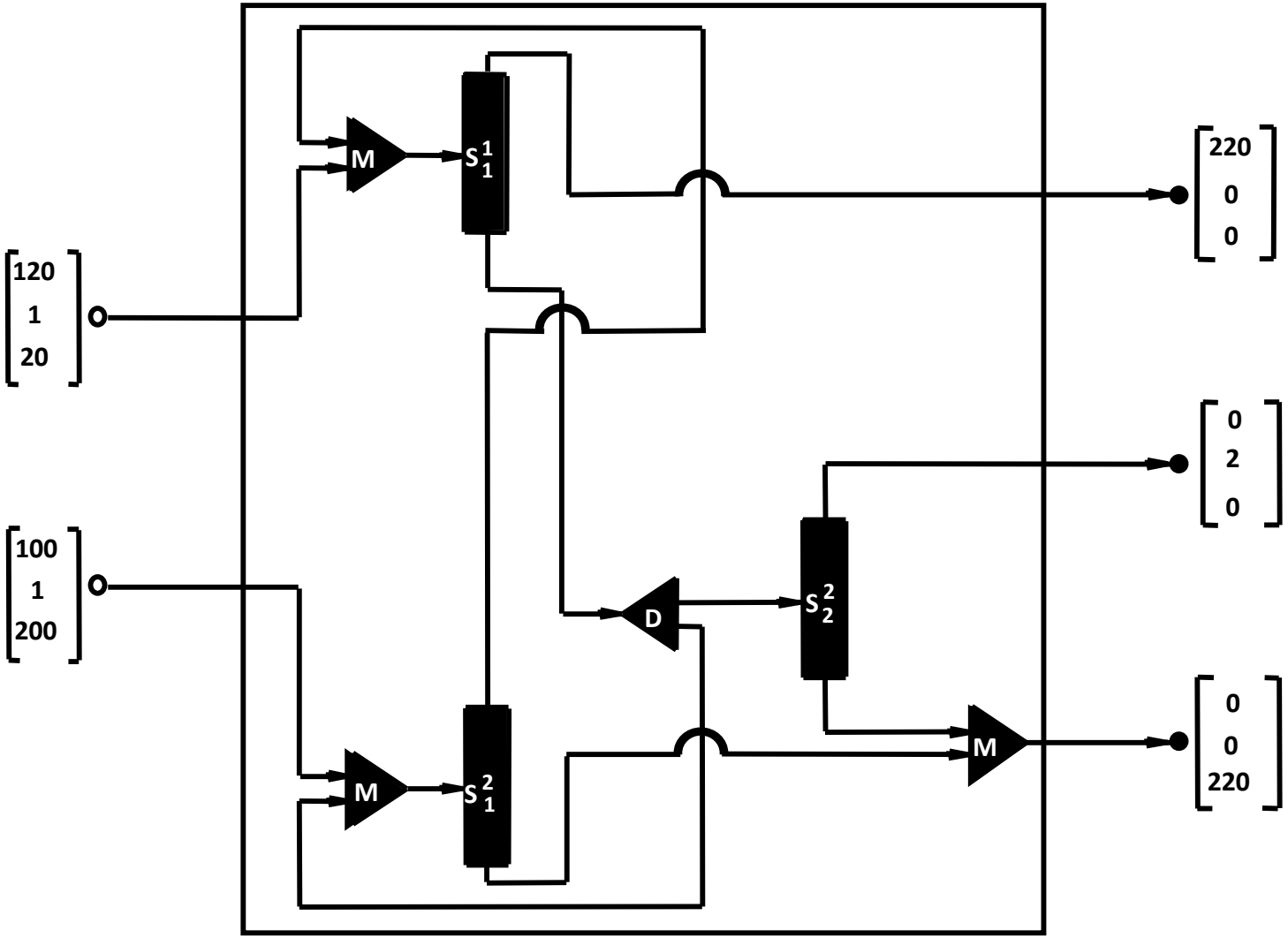


ILLUSTRATION OF THE DIFFICULTY OF SNS



UNEXPECTED PROPERTY: RECYCLING



Optimal network

SUMMARY OF STRUCTURAL PROPERTIES OF OPTIMAL SEPARATION NETWORKS

Product streams	Pure		Multicomponent	
Feed streams	Single	Multiple	Single	Multiple
recycling	impossible	possible	possible	possible
redundancy	impossible	possible	possible	possible
premixing	impossible	possible	impossible	possible
bypassing	impossible	impossible	possible*	possible *

* Maximal bypass is not necessary optimal

PROVED STATEMENTS ON THE STRUCTURAL PROPERTIES OF OPTIMAL SEPARATION NETWORKS

- If a network producing pure product-streams is optimal, each of its dividers must be in a loop of this network (concave, strictly monotone increasing, zero for zero mass-load cost function).
- An optimal separation network with a linear cost function do not contain recycling.
- Optimal separation networks with a nonlinear cost function may include redundant separators (concave, strictly monotone increasing, zero for zero mass-load cost function).
- An optimal separation network with a linear cost function may contain non-maximal bypasses.

SNS WITH LINEAR COST FUNCTION

This type of SNS problems is examined by:

Floudas (1987)

Wehe and Westerberg (1987)

Quesada and Grossmann (1995)

- Available algorithmic methods are usually based on incomplete super-structures.
- Available algorithmic mathematical programming models
  - do not exploit specific features of the class of problems, and
  - have nonlinear (bilinear) constraints.

PROPOSED METHOD

Features of the new method:

- Based on a rigorous super-structure
- Exploits the combinatorial features of the class of problems
- Generates better solution
- Faster

ALGORITHM SNS-LMSG

It generates the rigorous super-structure.

ALGORITHM SNS-LMSG

- Step 1. (Initialization.)
  - Step 1.1. Let each feed-stream (raw material) be represented by a vertex. Let each product-stream be represented by a vertex.
  - Step 1.2. Assign a divider to every feed-stream and connect each feed-stream to the corresponding divider, and a mixer to each product-stream and connect the outlet stream from the mixer to the corresponding product-stream.

ALGORITHM SNS-LMSG

- Step 2. (Creating separators and establishing bypasses.)
  - Step 2.1. Create all types of separators, each of which performs a separation between any pair of components in the feed-stream into the divider, and connect an outlet stream from the divider to another separator.
  - Step 2.2. Connect the outlet-streams from the divider to the mixers for the product-streams if it is plausible.

ALGORITHM SNS-LMSG

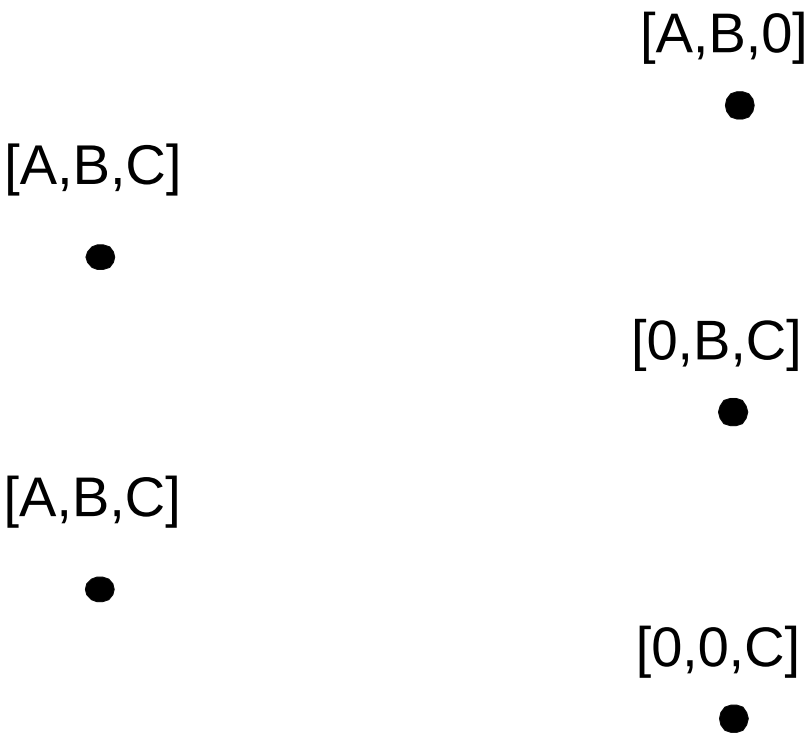
- Step 3. (Creating dividers.)

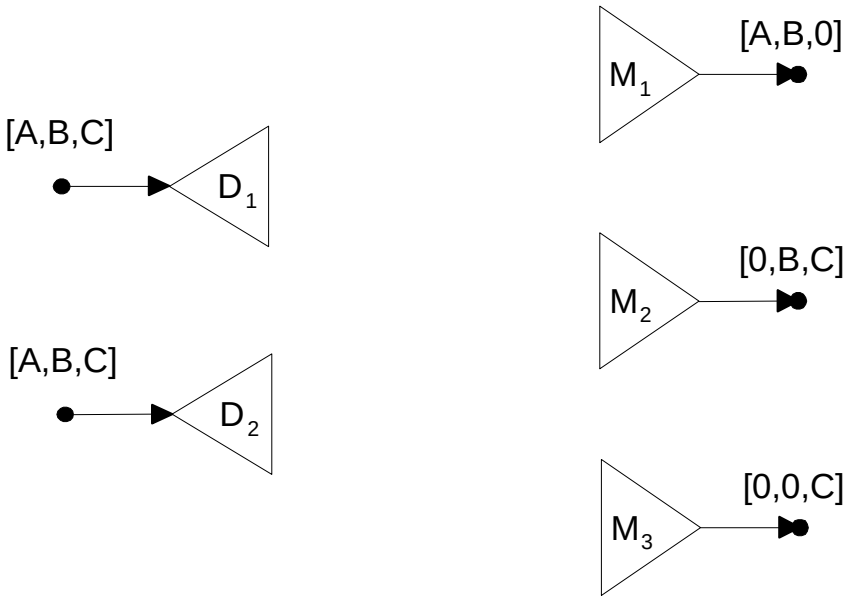
  Consider every separator in the structure.

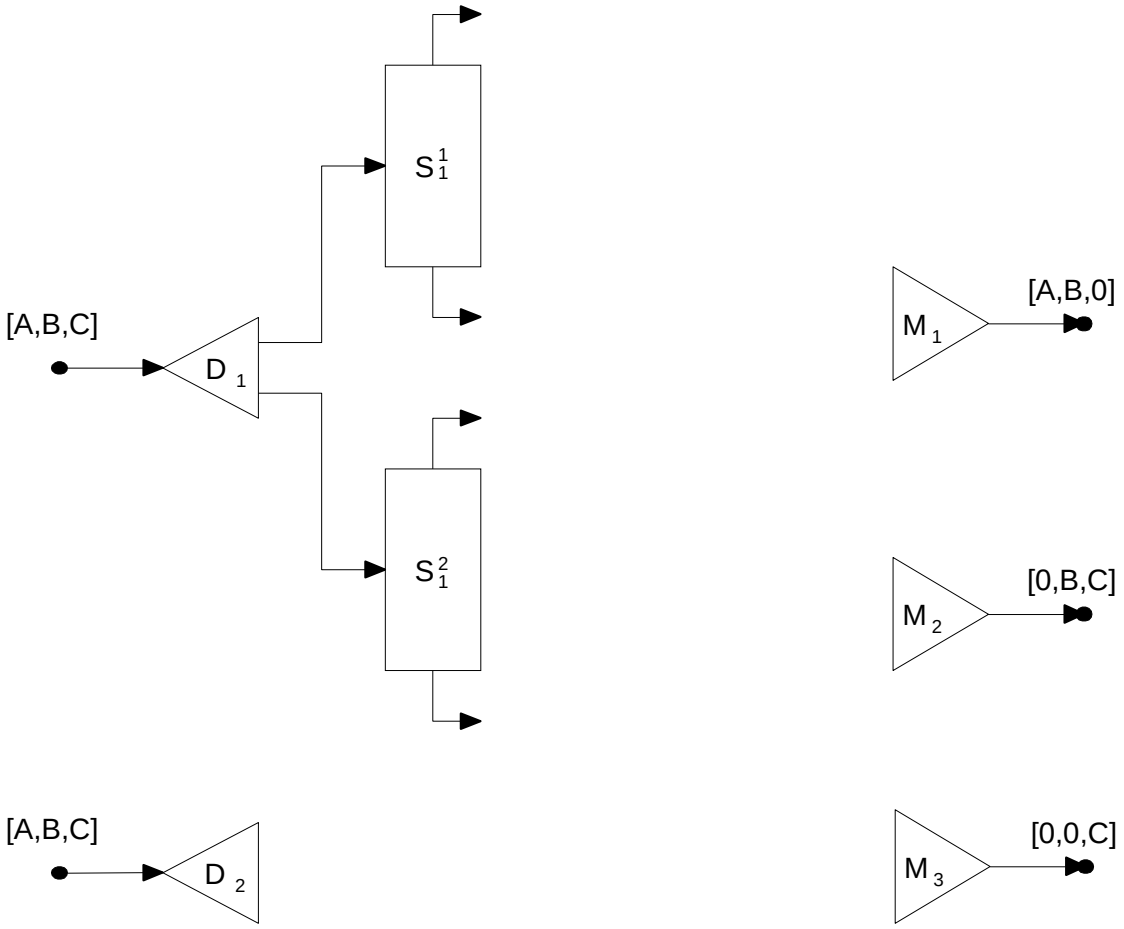
  - Step 3.1. Assign a divider to each outlet stream from the separator.
  - Step 3.2. Repeat Steps 2 and 3.

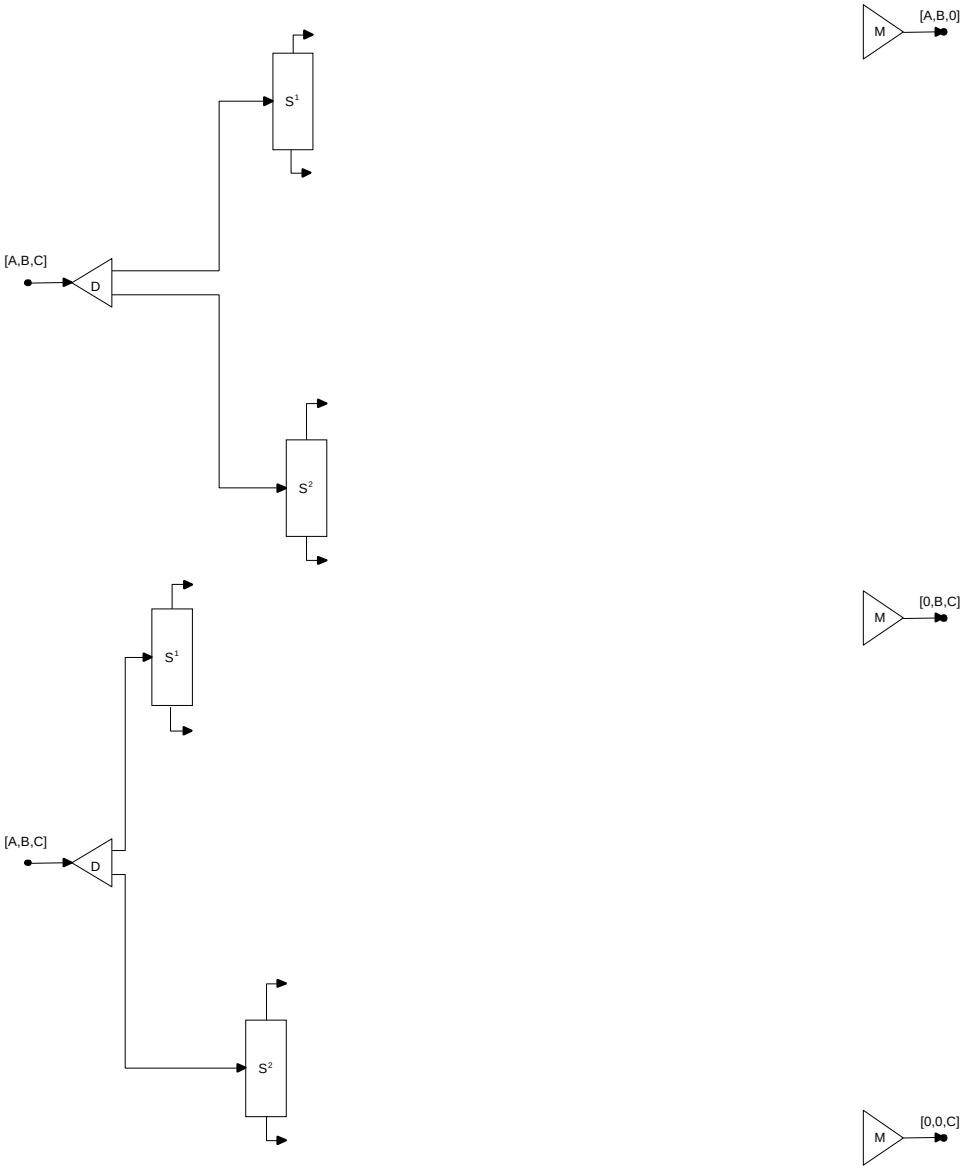
CLASS OF PROBLEMS TO ILLUSTRATE THE SUPER-STRUCTURE GENERATION

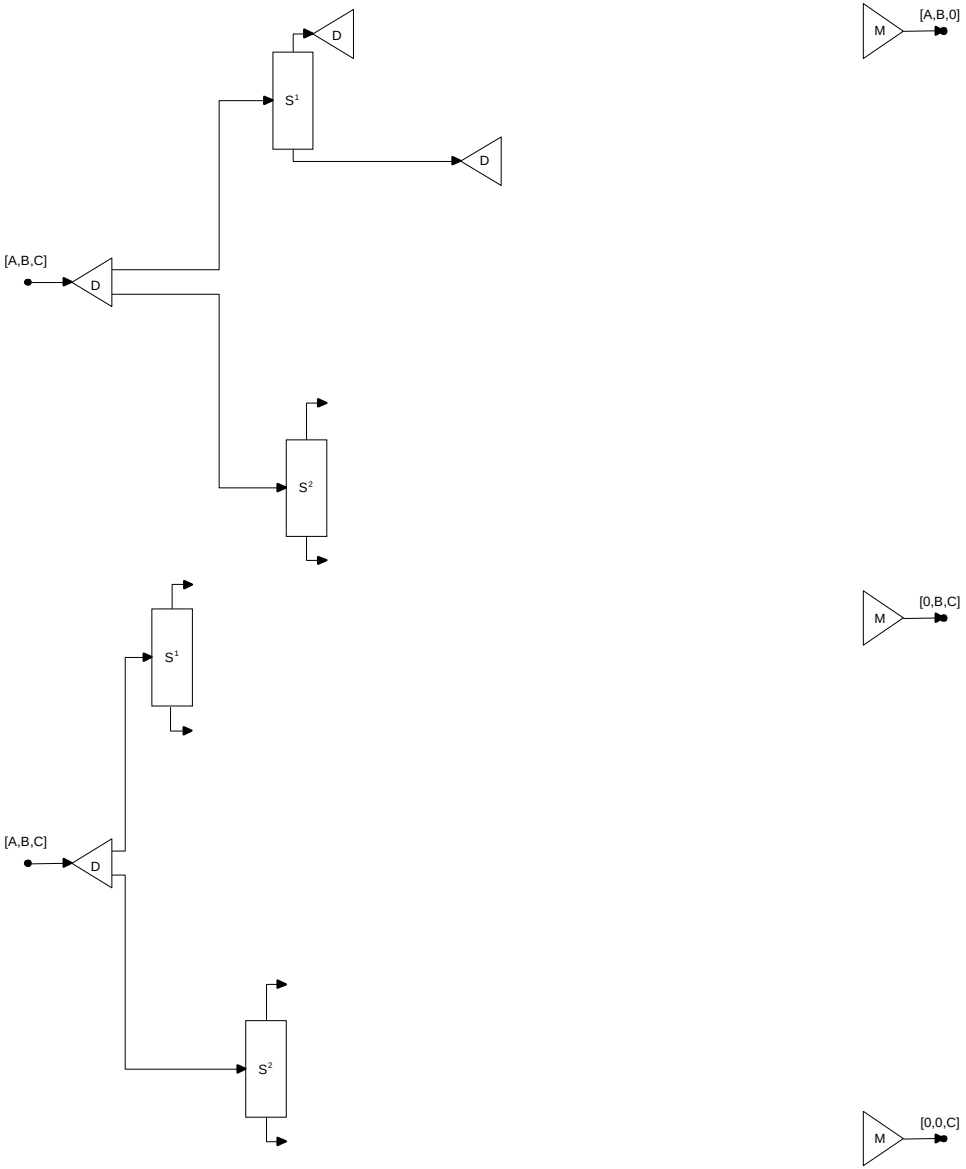
Stream	Component		
Feed-stream 1	A	B	C
Feed-stream 2	A	B	C
Product-stream 1	A	B	-
Product-stream 2	-	B	C
Product-stream 3	-	-	C

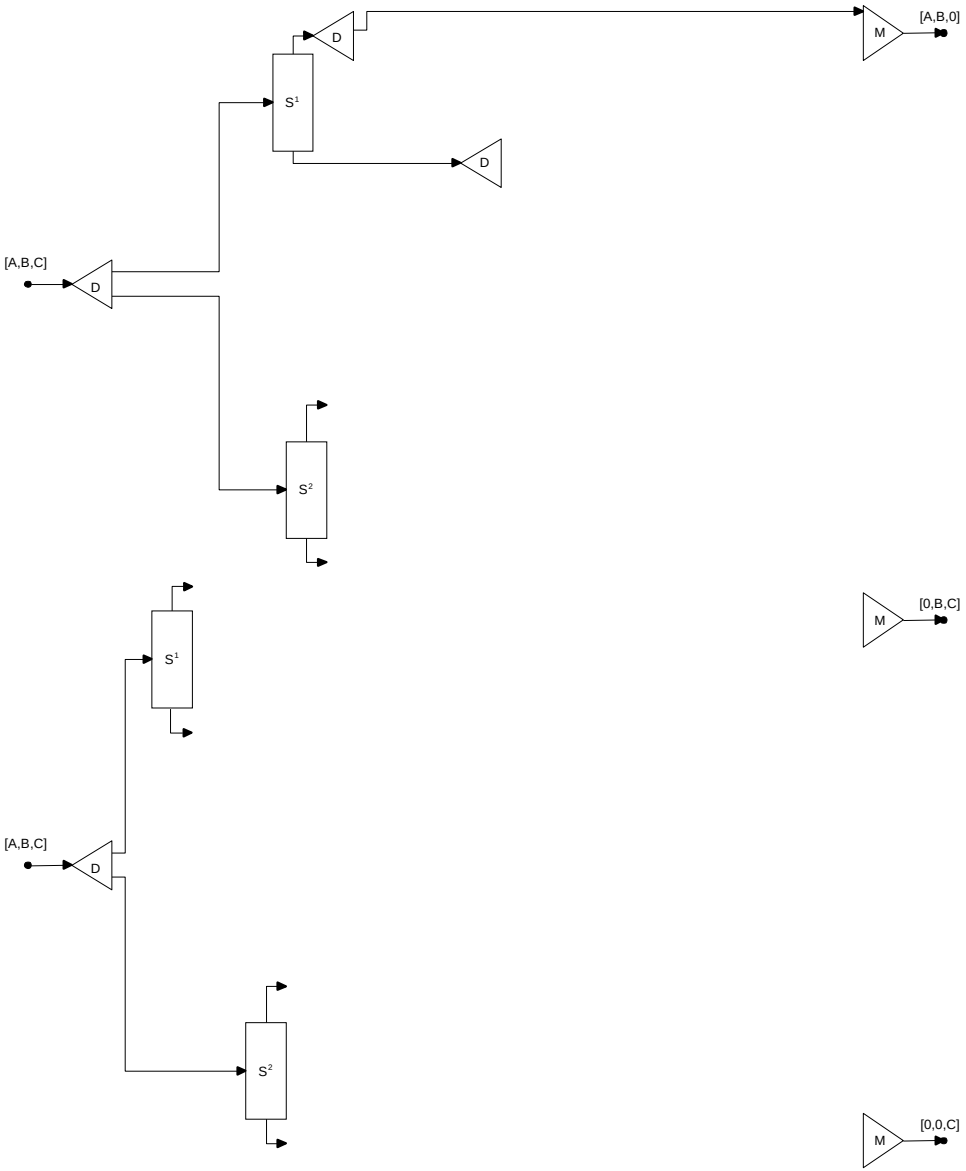

[A,B,0]
[A,B,C]
[0,B,C]
[A,B,C]
[0,0,C]

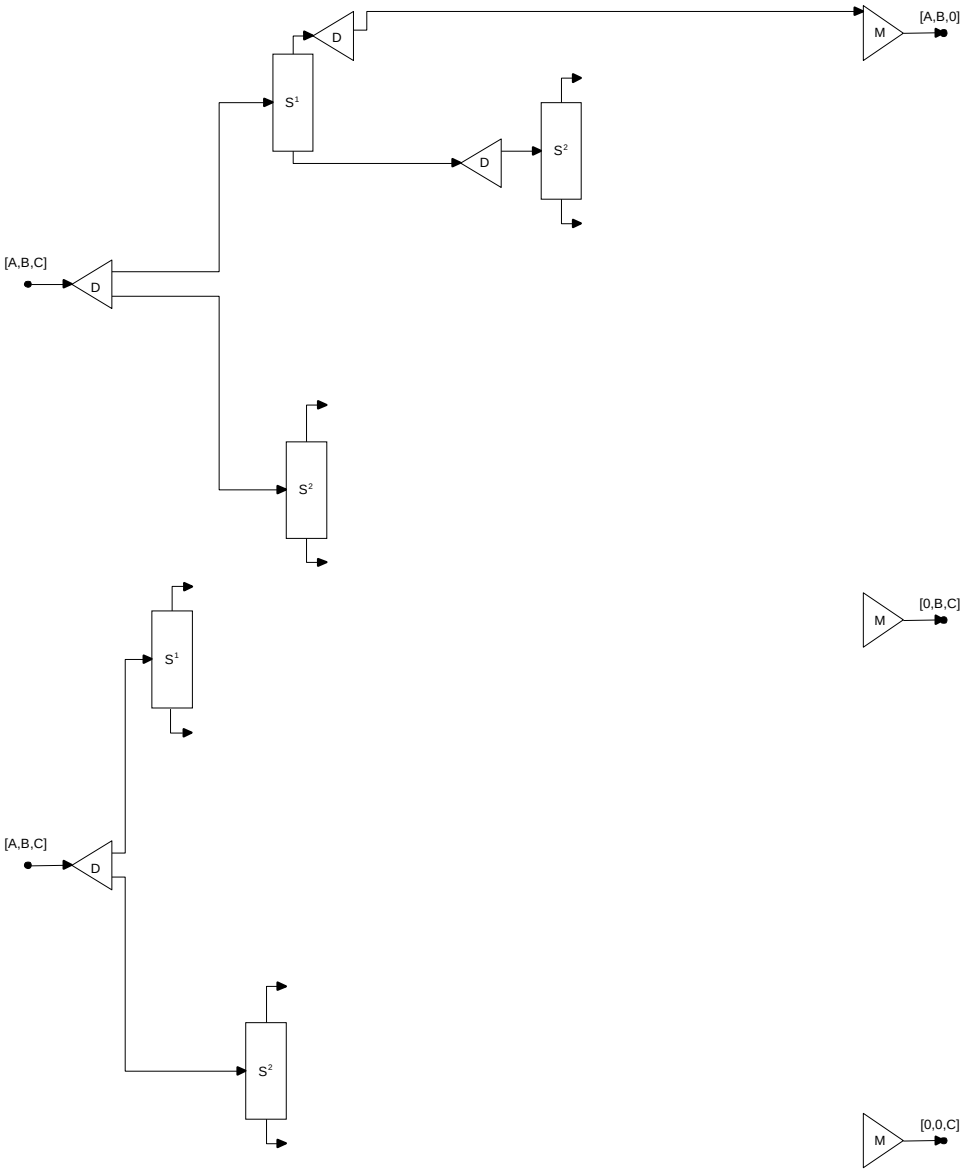

[A,B,C]
D1
[A,B,C]
D2
M1
[A,B,0]
M2
[0,B,C]
M3
[0,0,C]

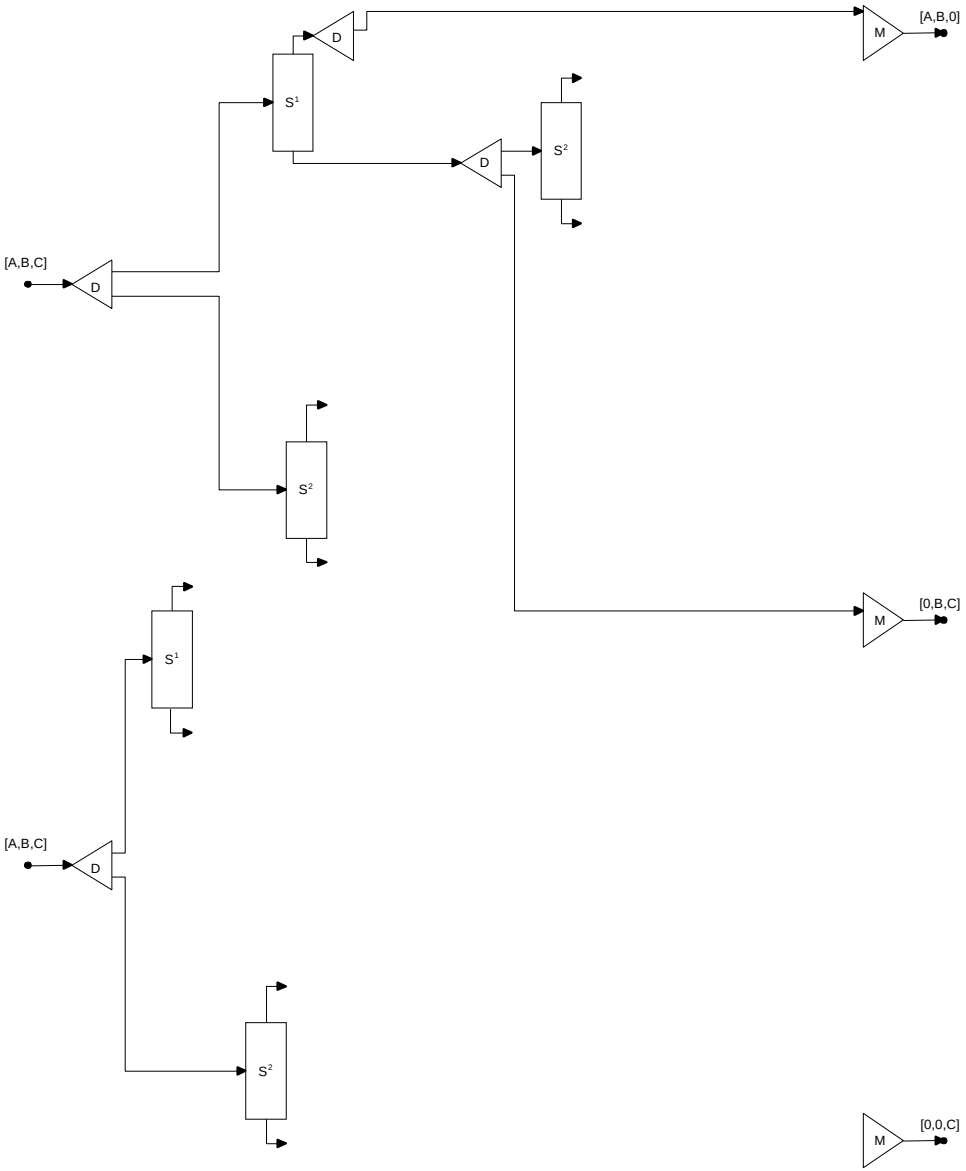

[A,B,C]
D 1
S 1 1
S 1 2
[A,B,C]
D 2
M 1
[A,B,0]
M 2
[0,B,C]
M 3
[0,0,C]

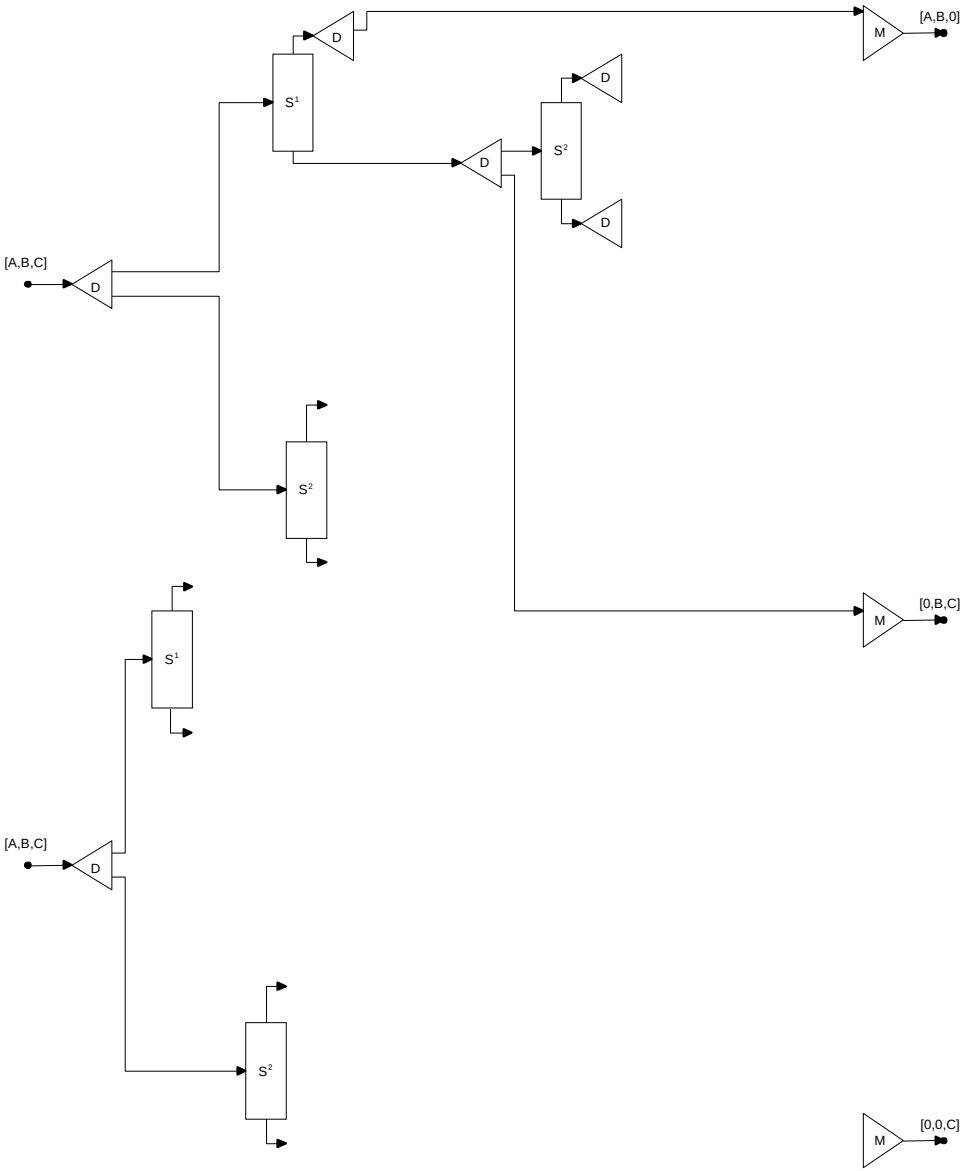

[A,B,C]
D
S1
S2
M
[A,B,0]
[0,B,C]
[0,0,C]

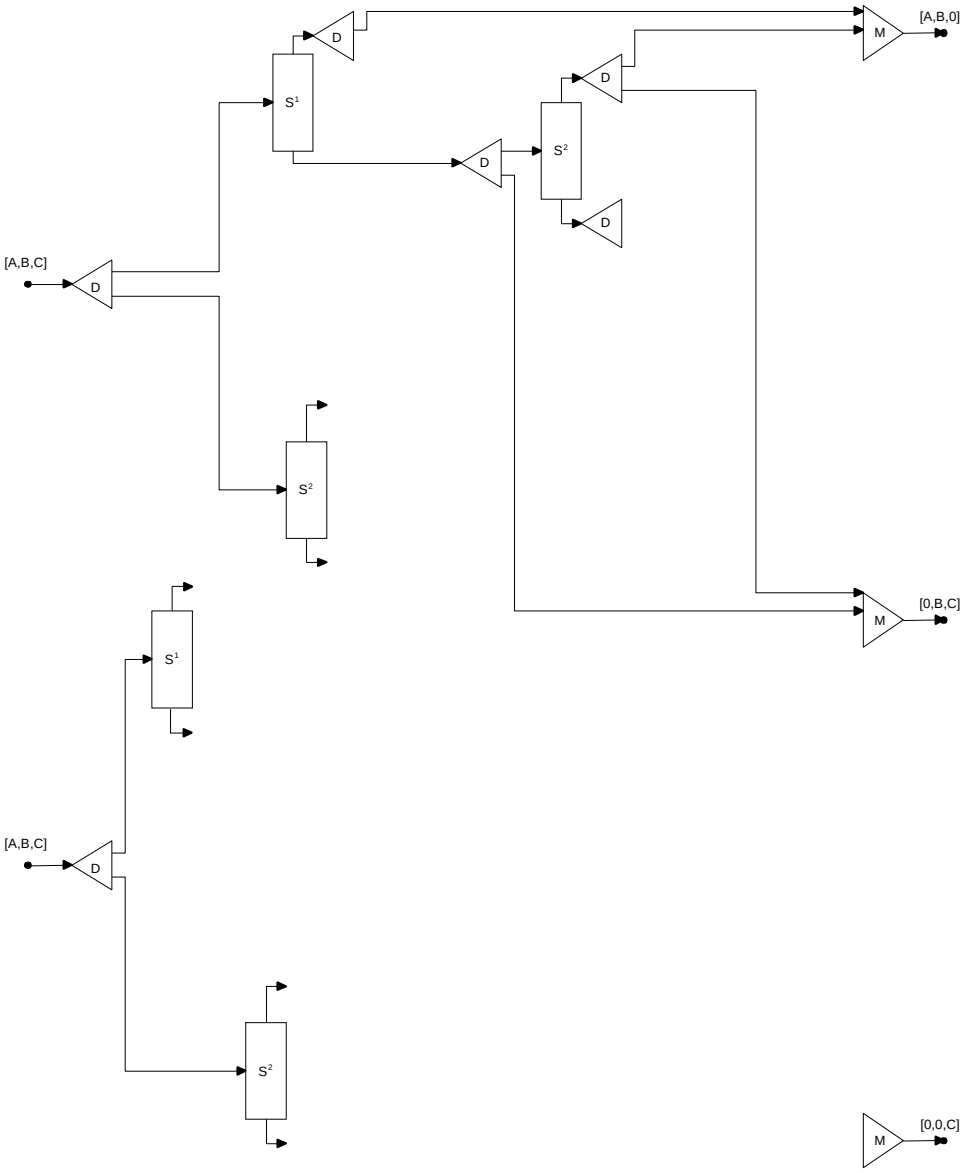

D
S1
D
[A,B,C]
D
S2
M
[A,B,0]
S1
[A,B,C]
D
S2
M
[0,B,C]
M
[0,0,C]

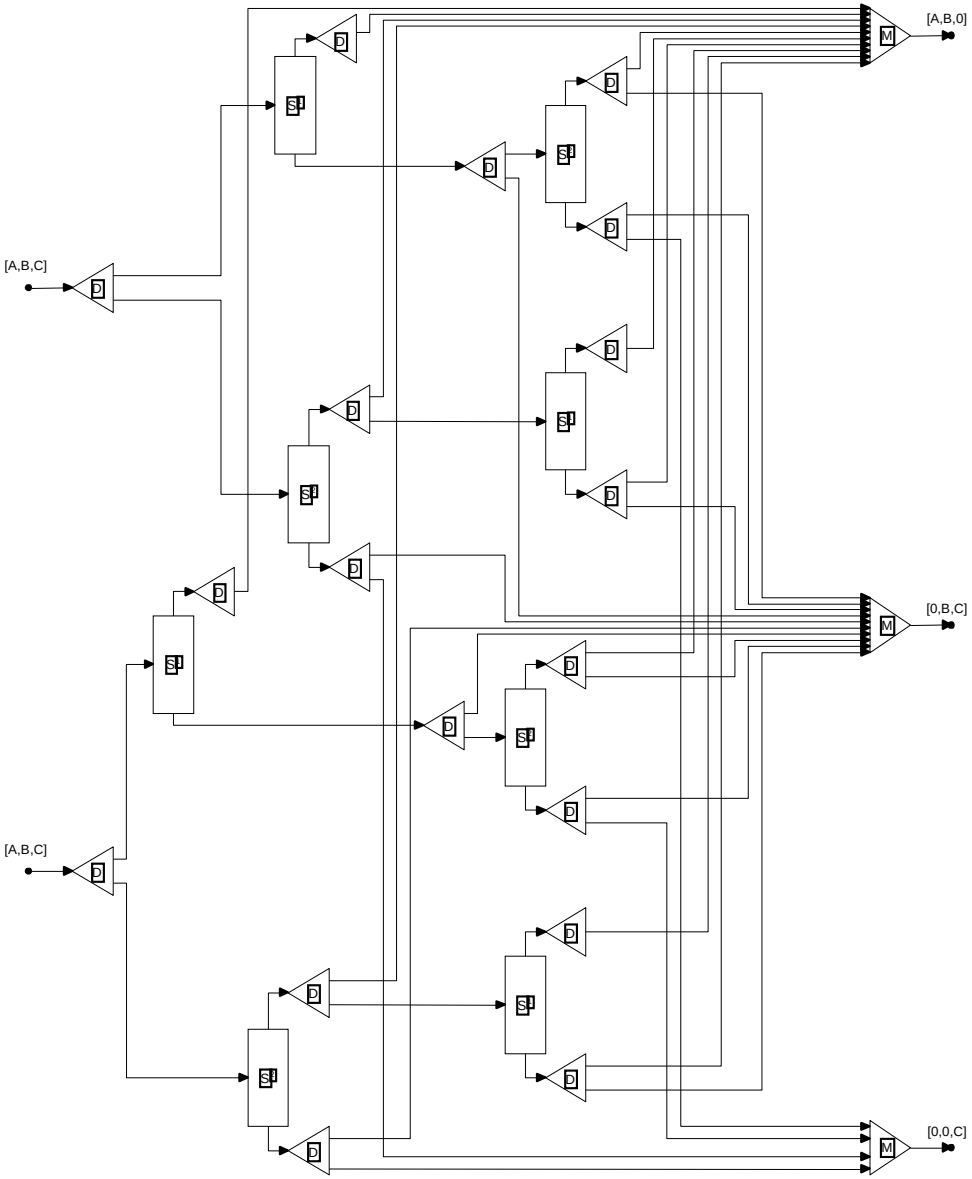

D
M
[A,B,0]
S¹
D
[A,B,C]
D
S²
S¹
M
[0,B,C]
[A,B,C]
D
S²
M
[0,0,C]


[A,B,C]
D
S¹
S²
D
D
M
[A,B,0]
S²
S¹
S²
M
[0,B,C]
M
[0,0,C]


[A,B,C]
D
S1
S2
D
D
M
[A,B,0]
S2
M
[0,B,C]
[A,B,C]
D
S1
S2
M
[0,0,C]


[A,B,C]
D
S¹
S²
D
D
D
D
M
[A,B,0]
S²
S¹
M
[0,B,C]
[A,B,C]
D
S²
M
[0,0,C]


[A,B,C]
D
S¹
S²
D
D
D
D
M
[A,B,0]
S²
M
[0,B,C]
S¹
[A,B,C]
D
S²
M
[0,0,C]


[A,B,0]
[A,B,C]
[0,B,C]
[A,B,C]
[0,0,C]

MATHEMATICAL MODEL BASED ON RIGOROUS SUPER-STRUCTURE

- Generated algorithmically
- LP

EXAMPLE SNS 1 (Quesada and Grossmann, 1995)

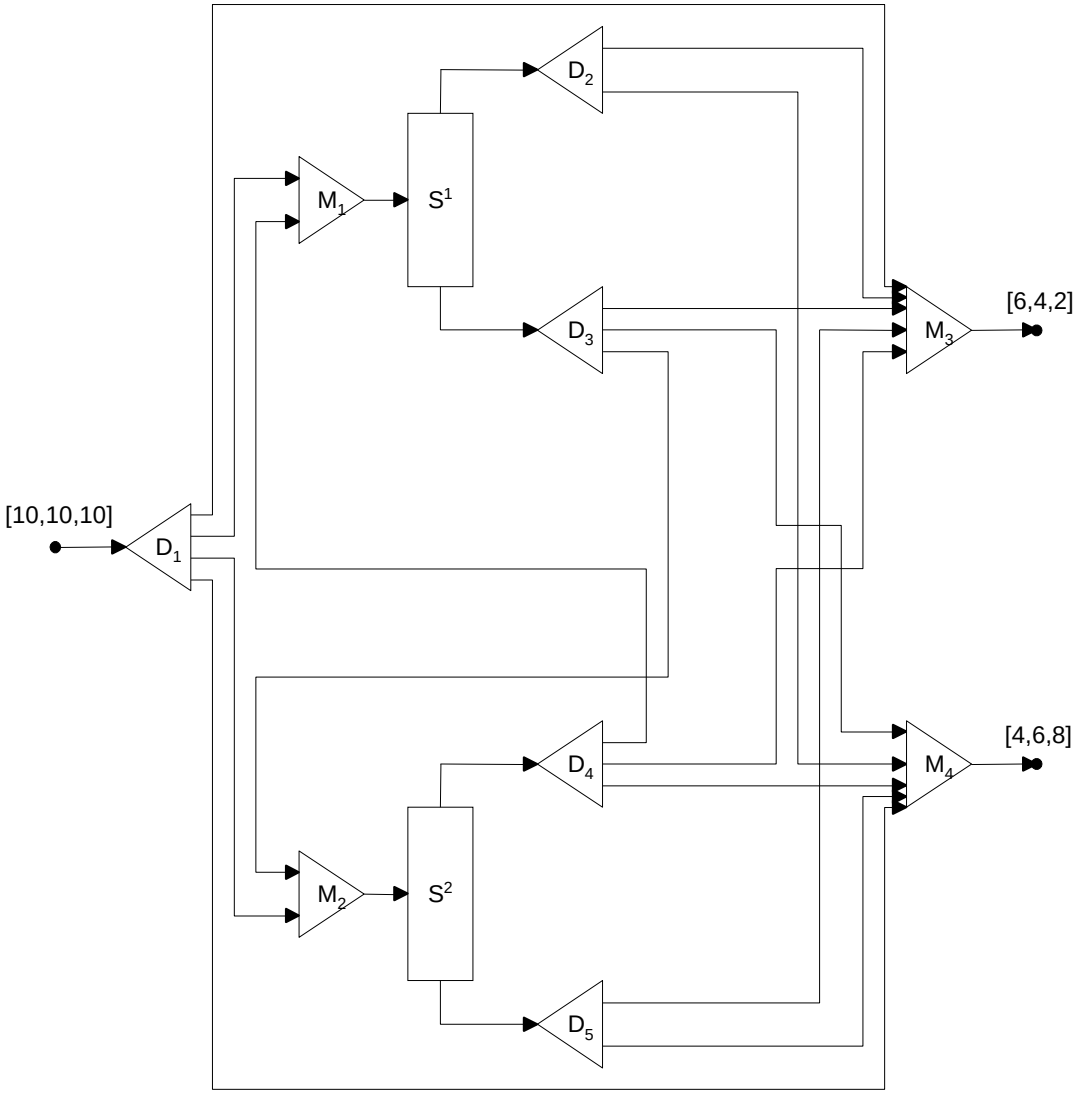
Component	A	B	C
Feed-stream	10	10	10
Product-stream 1	6	4	2
Product-stream 2	4	6	8

The objective function to be minimized is the sum of the total flows into the separators.

Super-structure in

Quesada and

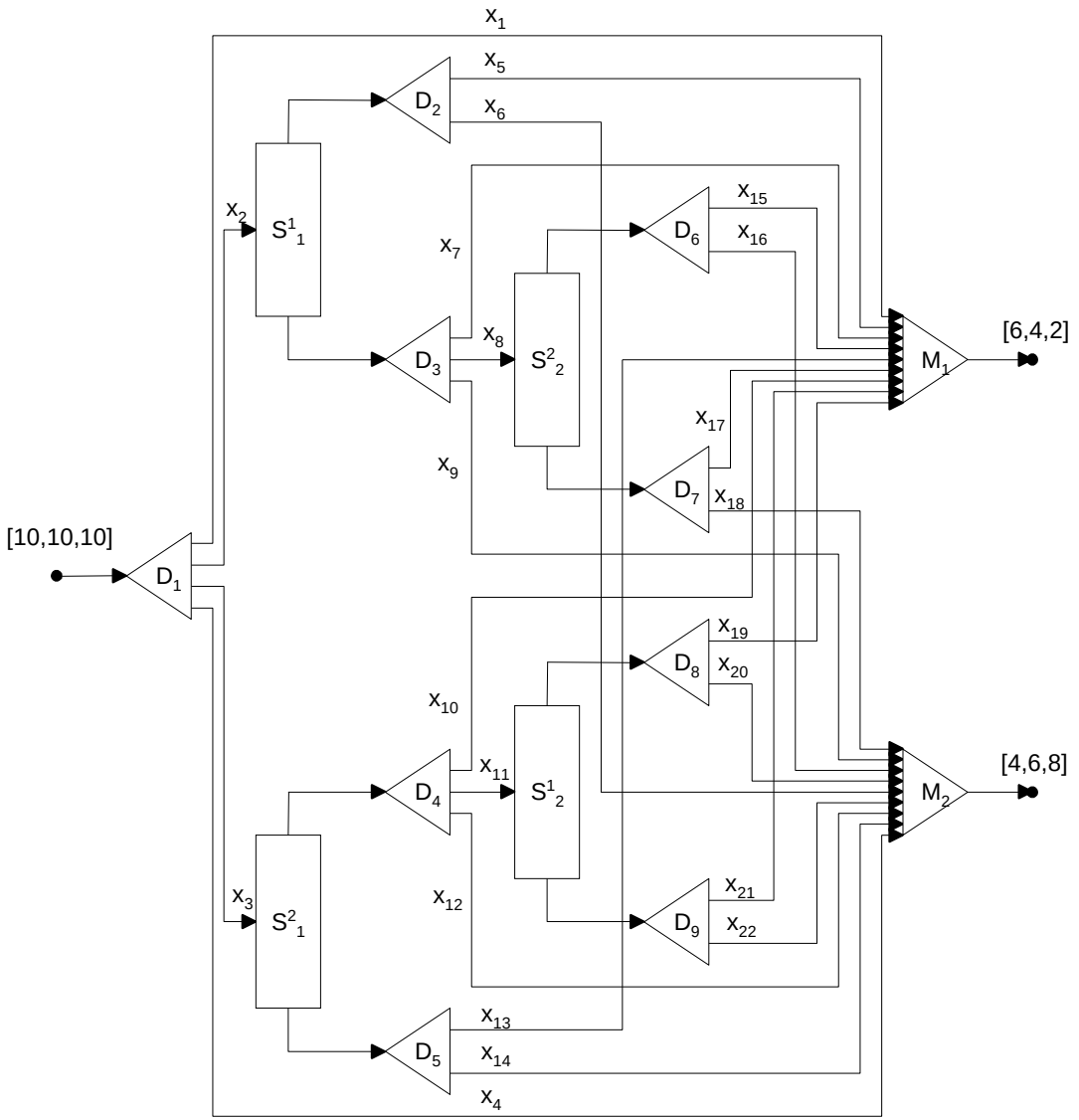
Grossmann (1995)



Rigorous

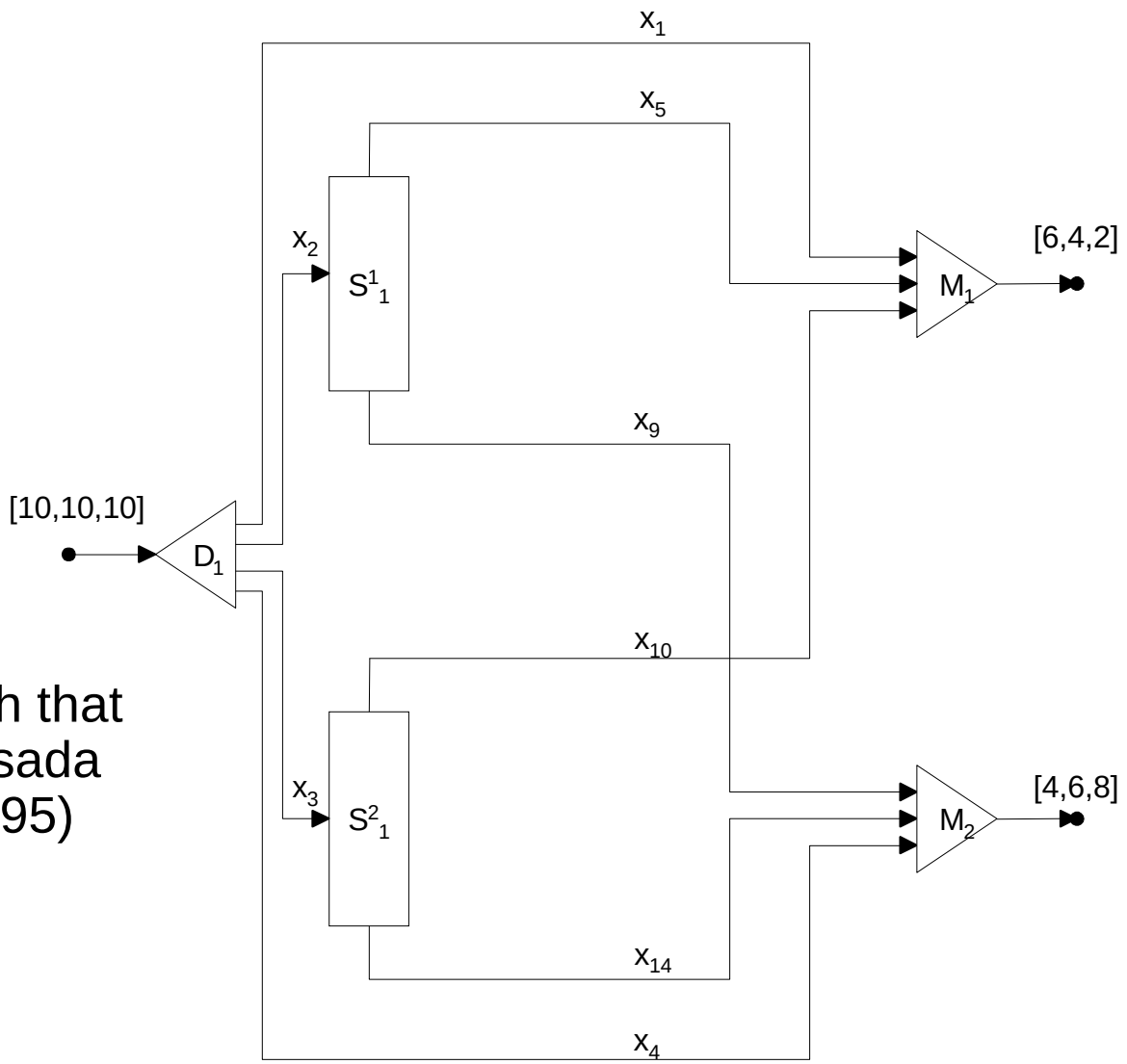
super-
structure

(generated)



Optimal solution

Optimum: 12

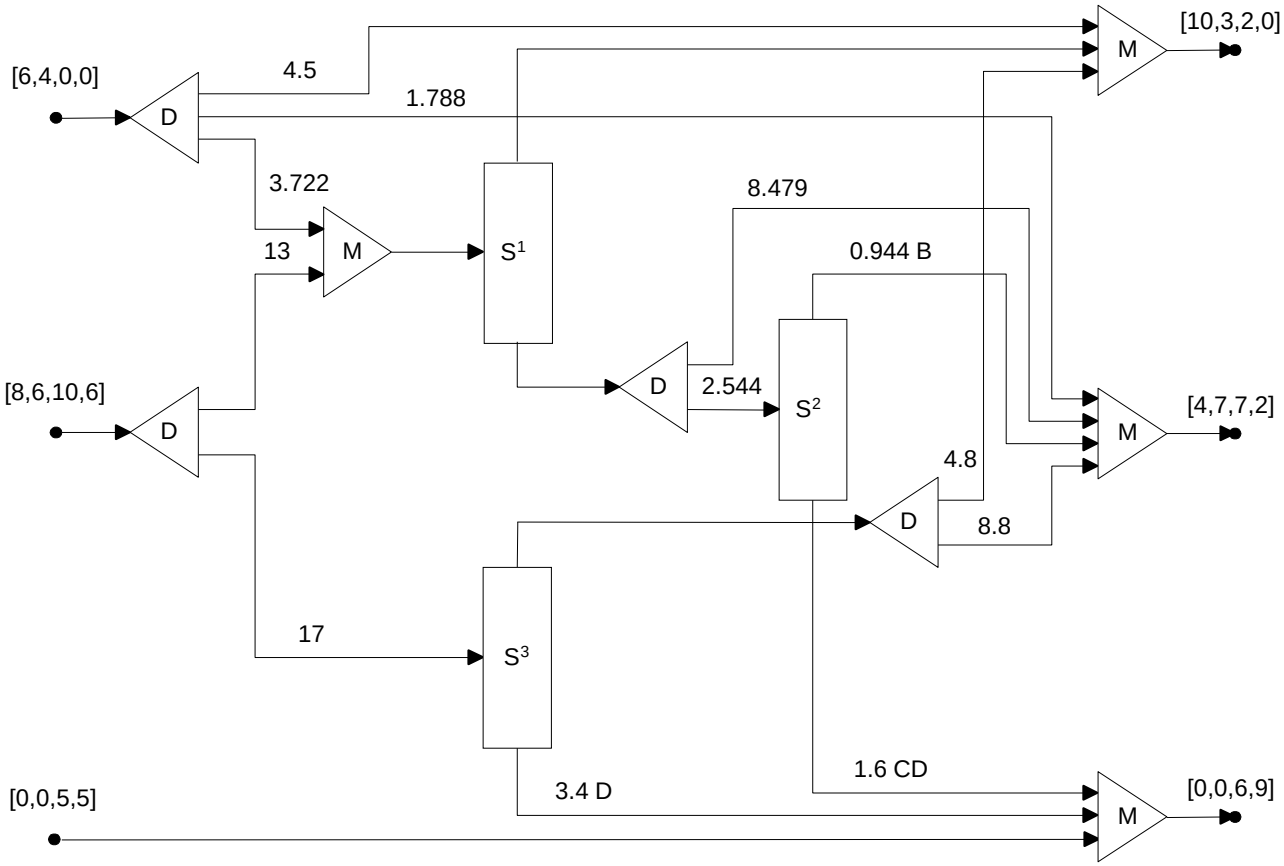


Note: it is identical with that of published in Quesada and Grossmann (1995)

EXAMPLE SNS 2 (Quesada and Grossmann, 1995)

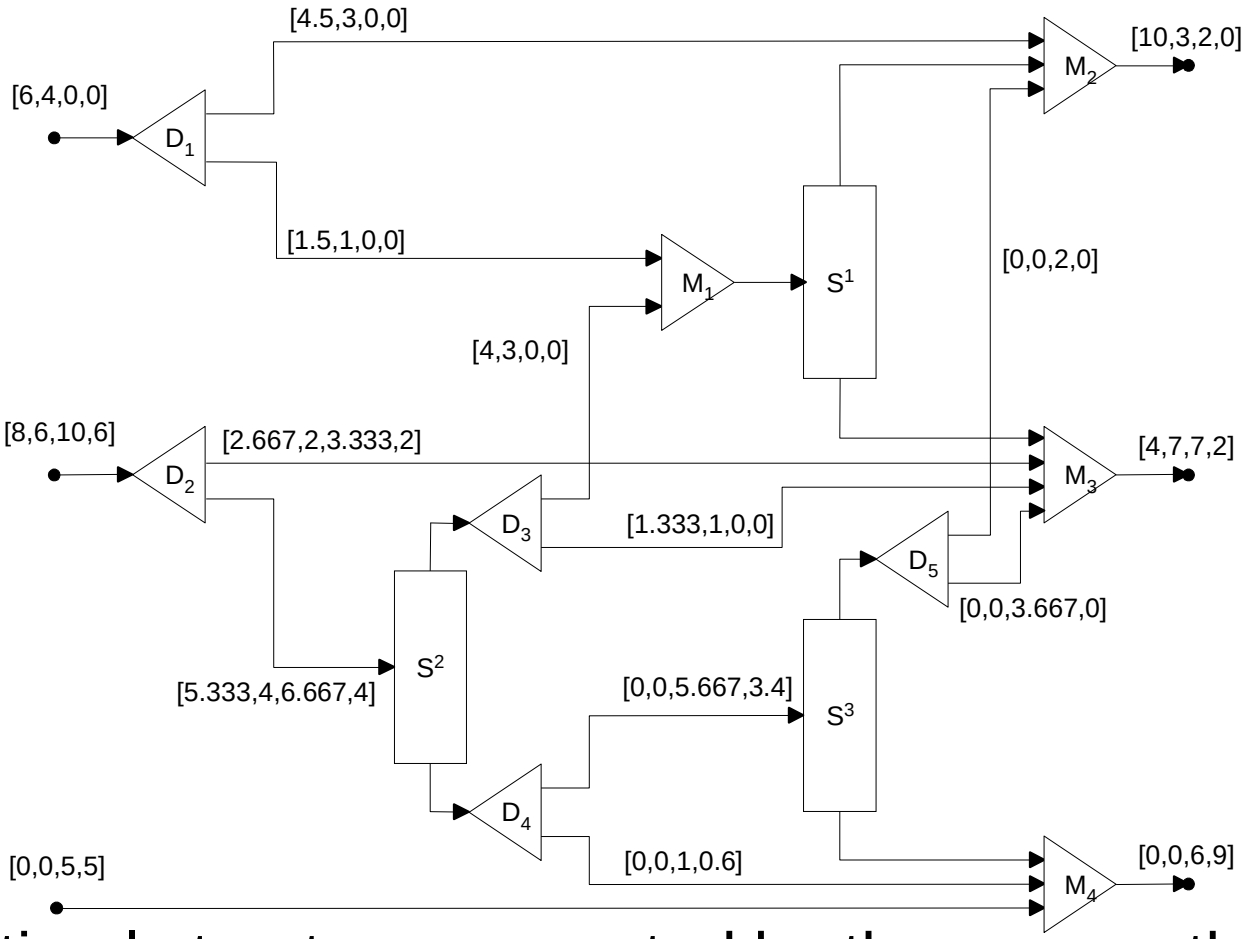
Component	A	B	C	D
Feed-stream 1	6	4	0	0
Feed-stream 2	8	6	10	6
Feed-stream 3	0	0	5	5
Degree of Difficulty	4	1.5	4	

Product	Sum of the components	Component Information			
Product-stream 1	15	A≥9	B≤3	C≤3	D=0
Product-stream 2	20	B≥7	C≥7	B=C	
Product-stream 3	15	D≥9	A=0		



Structure obtained by Quesada and Grossmann (1995)

The value of the cost function is 138.7.



Optimal structure generated by the new method:

The value of the cost function is 104.3.

COMPARISON IN SOLVING EXAMPLE SNS 2

Method based on	Number of variables	Type of model	Optimal solution	Computation time
Quesada and Grossmann's super-structure	113	nonlinear	138.7	0.74*
Proposed rigorous super-structure	90	linear	104.3	0.55**

* From the publication: on IBM RS600/530.
Note that we were not able to duplicate the result.

** On a PC (Pentium, 100MHz) with GAMS as the solver.

CONCLUDING REMARKS ON SEPARATION NETWORK SYNTHESIS

This simple class of synthesis problems illustrates:

- the difficulty of synthesis
- the need for mathematical foundation
- algorithmic solution (optimality guaranteed)

COMBINATORIAL TECHNIQUE IN PROCESS DESIGN AND SYNTHESIS

INTRODUCTION

MINLP

$$\min g(x,y)$$

s.t.

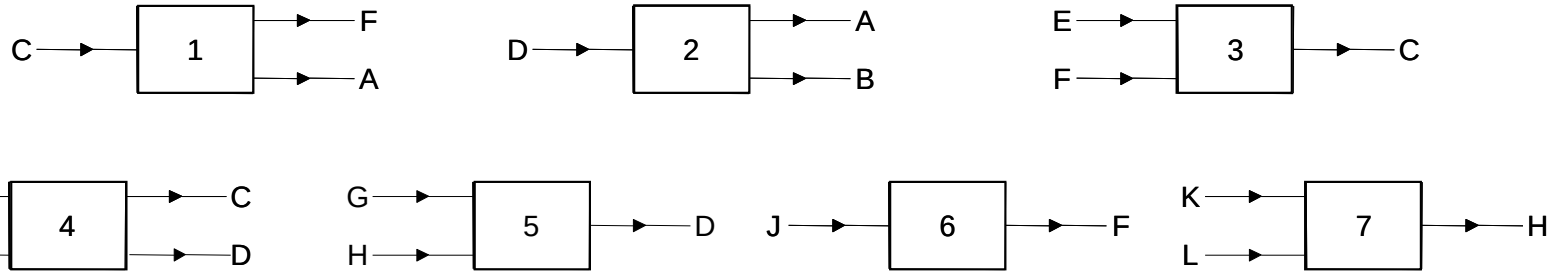
$$f(x, y) \leq 0$$

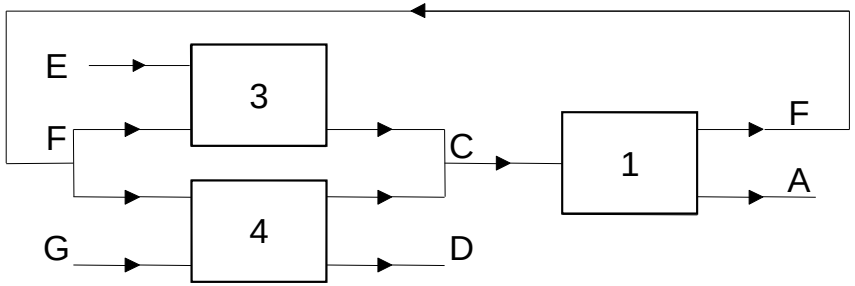
$$x \in \Re^n,\ y \in \{0, 1\}^m$$

- Most MINLP model can not represent a practical problem.
- Additional information is embedded implicitly in the model of a practical problem.
- Idea: this information can effectively control the procedure.

ILLUSTRATIVE EXAMPLE PNS 1

Operating units:





Product: A

Raw materials: E, G, J, K, L

Feasible flowsheet

EXAMPLE PNS 1

Product: A

Raw materials: E, G, J, K, L

Plausible operating units

Type	Inputs	Outputs
1	C	A, F
2	D	A, B
3	E, F	C
4	F, G	C, D
5	G, H	D
6	J	F
7	K, L	H

Number of

operating units:	7	
binary variables:	7	
combinations:	127	$(=2^7-1)$

SYNTHESIS OF AN INDUSTRIAL PROCESS (EXAMPLE PNS 2)

Product: A61

Raw materials: A1, A2, A3, A4, A6, A7, A8, A11, A15, A17, A18, A19, A20, A23, A27, A28, A29, A30, A34, A43, A47, A49, A52, A54

PLAUSIBLE OPERATING UNITS

No.	Type	Inputs	Outputs
1	Feeder	A1	A5
2	Reactor	A2, A3, A4	A9
3	Reactor	A3, A4, A6, A11	A10
4	Reactor	A3, A4, A5	A12
5	Reactor	A3, A4, A5	A13
6	Reactor	A7, A8, A14	A16
7	Reactor	A8, A14, A18	A16
8	Separator	A9, A11	A21, A22, A24
9	Separator	A10, A11	A22, A24, A37
10	Separator	A12	A25, A26
11	Separator	A13	A25, A31
12	Dissolver	A15, A16	A32

PLAUSIBLE OPERATING UNITS (Cont'd)

No.	Type	Inputs	Outputs
13	Reactor	A14, A17, A18, A19, A20	A33
14	Reactor	A6, A21	A35
15	Washer	A22, A23	A48
16	Washer	A5, A24	A36
17	Separator	A5, A11, A25	A37, A38, A39
18	Separator	A11, A26	A40, A42
19	Reactor	A14, A27, A28, A29, A30	A41
20	Separator	A11, A31	A40, A42
21	Centrifuge	A32	A44, A45
22	Washer	A33, A34	A46
23	Separator	A36	A14, A48
24	Separator	A38	A14, A48

PLAUSIBLE OPERATING UNITS (Cont'd)

No.	Type	Inputs	Outputs
25	Filter	A41	A50, A51
26	Washer	A43, A44	A53
27	Filter	A46	A55, A56
28	Separator	A47, A48	A5, A57
29	Separator	A48, A49	A5, A58
30	Separator	A50	A59, A60
31	Dryer	A51, A54	A61
32	Dryer	A52, A53	A61
33	Dryer	A54, A55	A61
34	Distillation	A59	A62, A63
35	Separator	A60	A64, A65

Number of

operating unit: 35

binary variables: 35

combinations: 34 billion

subproblems at a B&B (worst case): 130 million

SOURCE OF COMPLEXITY

Combinatorial nature of the problem

COMBINATORIAL TOOLS

Our rigorous technique is based on combinatorics, especially,

on the following items.

- P-graph

  New structure representation.

- Axioms

  The fundamental properties of combinatorially feasible process structures (e.g., every operating unit has at least one path leading to a product).

- Algorithms

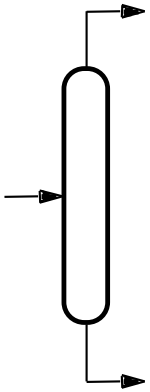
  Effective and rigorous combinatorial algorithms for process synthesis.

STRUCTURAL REPRESENTATION

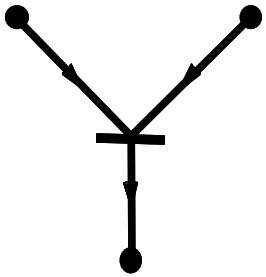
- Simple directed graphs are incapable of providing an unambiguous representation in process synthesis.
- Process graphs or P-graphs are introduced for structural representation in process synthesis.

CONVENTIONAL AND P-GRAPH REPRESENTATION

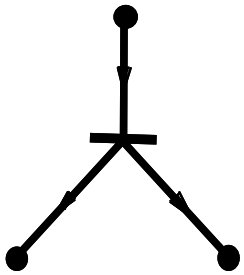
reactor



distillation column



reactor



distillation column

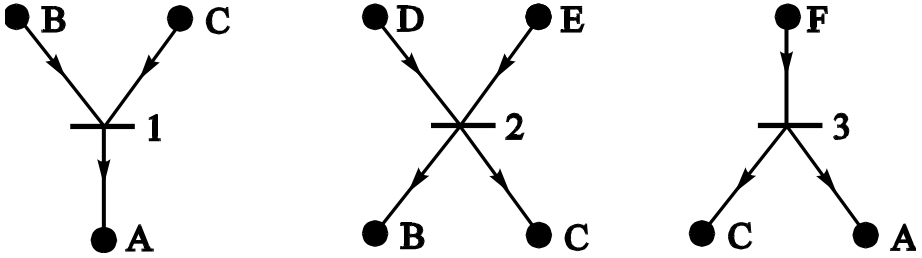
P-graph

FORMAL DEFINITION: P-GRAPH

- A P-graph can be considered as a directed bipartite graph.
  M is the set of materials
  O is the set of operating units, where
  $O \subseteq \wp(M) \times \wp(M)$, $O \cap M = \varnothing$
- If $(\alpha,\beta) \in O$, then, α is the input set, and β is the output set of this operating unit.
- Pair (M,O) is defined to be a P-graph with the set of vertices $M \cup O$ and the set of arcs

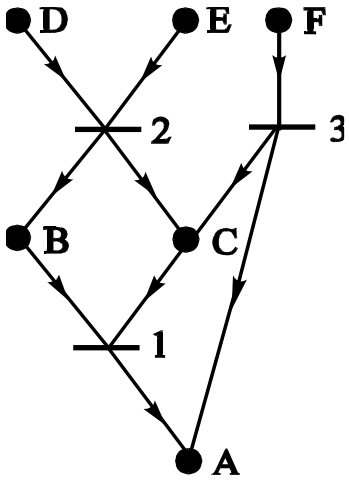
  $\{(x,y): y=(\alpha,\beta) \in O \ \& \ x \in \alpha\} \cup \{(y,x): y=(\alpha,\beta) \in O \ \& \ x \in \beta\}$.

Example



M1={A, B, C, D, E, F}

O1={({B, C}, {A}), ({D, E}, {B, C}), ({F}, {A, C})}



P-graph (M1, O1)

FORMAL DESCRIPTION OF THE COMBINATORIAL COMPONENT OF PNS

- Let a finite set of materials M be given.
- The combinatorial components of a PNS problem is given by triplet (P,R,O)

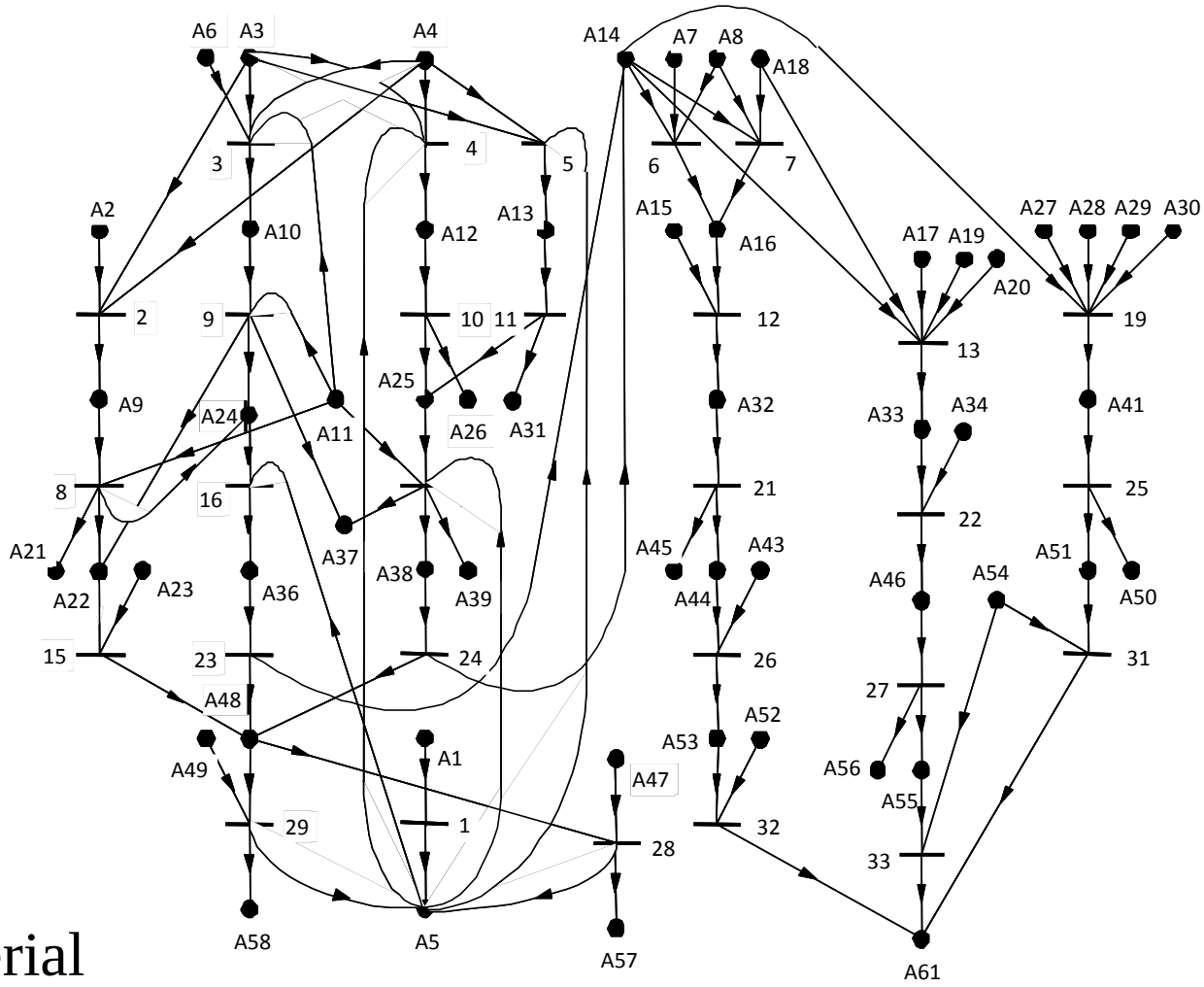
  where

  $P \subseteq M$ is the set of products to be produced

  $R \subseteq M$ is the set of raw materials

  $O \subseteq \wp(M) \times \wp(M)$ is the set of operating units.
- It is assumed that $P \cap R = \varnothing$.

P-GRAPH REPRESENTATION OF A SYNTHESIS PROBLEM PNS 2



Notation:

- material
- operating unit

AXIOMS OF COMBINATORIALY FEASIBLE PROCESS STRUCTURES

For given process synthesis problem, a P-graph satisfying the following five axioms is a combinatorially feasible structure.

(S1) Every final product is represented in the structure.

(S2) A material represented in the structure is a raw material if and only if it is not an output of any operating unit represented in the structure.

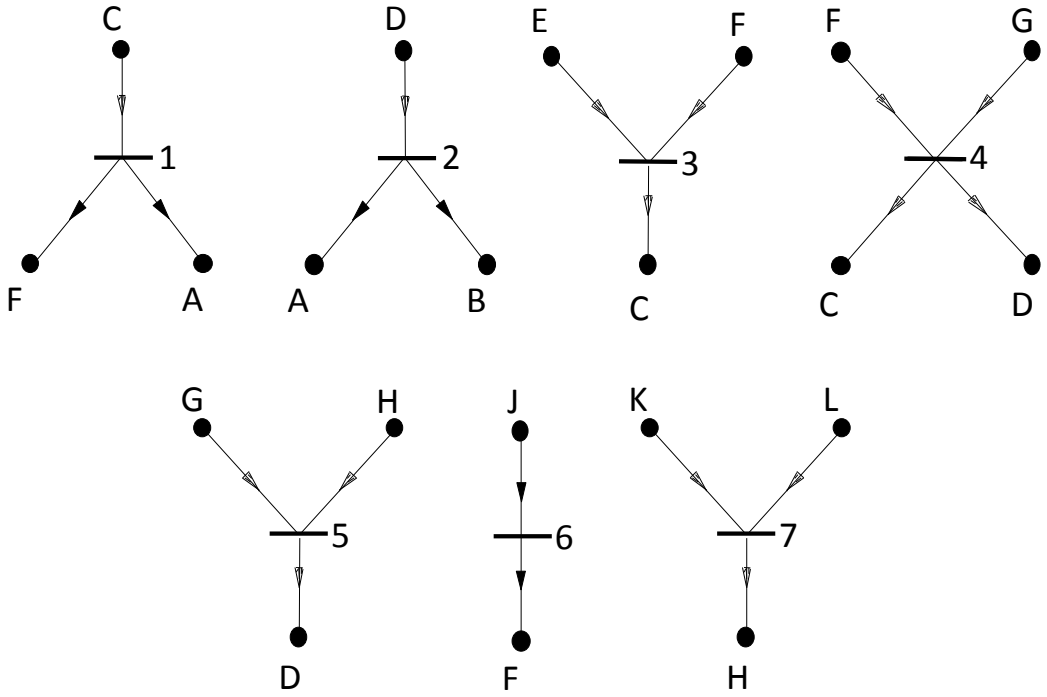
(S3) Every operating unit represented in the structure is defined in the synthesis problem.

(S4) Any operating unit represented in the structure has at least one path leading to a product.

(S5) If a material belongs to the structure, it must be an input to or output from at least one operating unit represented in the structure.

ILLUSTRATIVE EXAMPLE FOR THE COMBINATORIALLY FEASIBLE STRUCTURES: EXAMPLE PNS 1

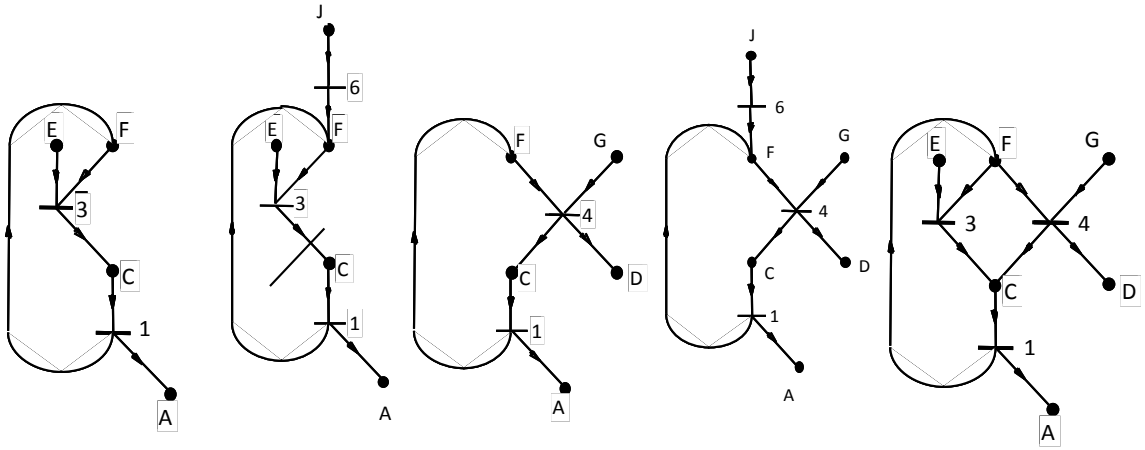
Operating units given:



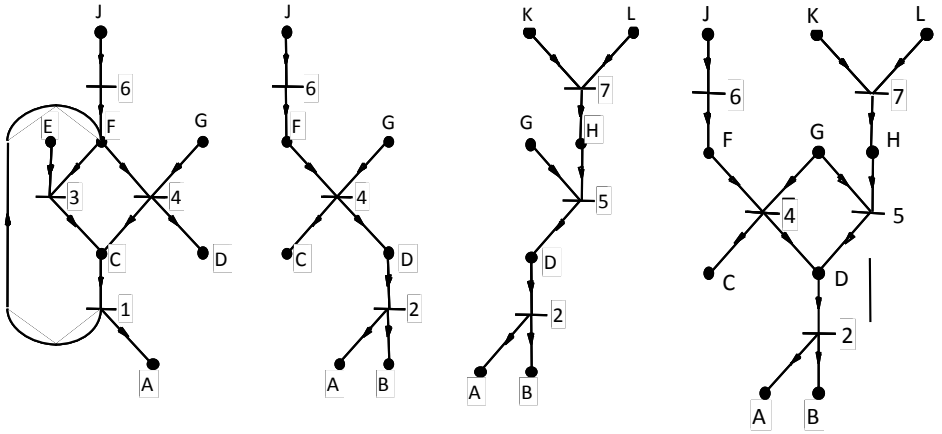
Available raw materials: E, G, J, K, L

Product: A

COMBINATORIALLY FEASIBLE STRUCTURES OF EXAMPLE PNS 1

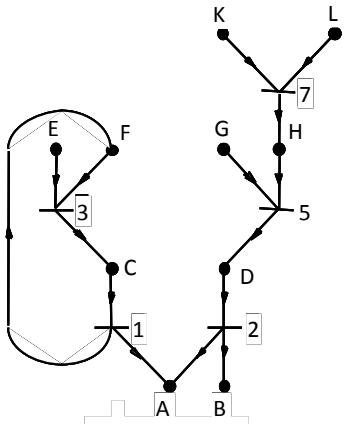


Solution #1 Solution #2 Solution #3 Solution #4 Solution #5

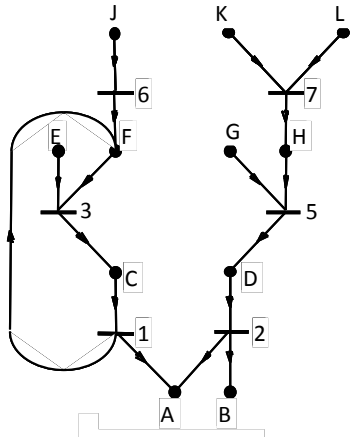


Solution #6 Solution #7 Solution #8 Solution #9

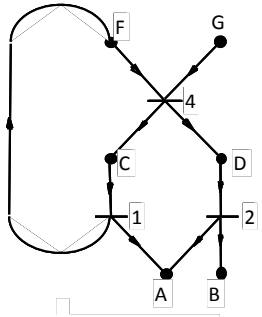
COMBINATORIALLY FEASIBLE STRUCTURES OF EXAMPLE PNS 1 (Cont'd)



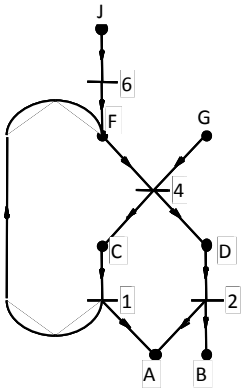
Solution #10



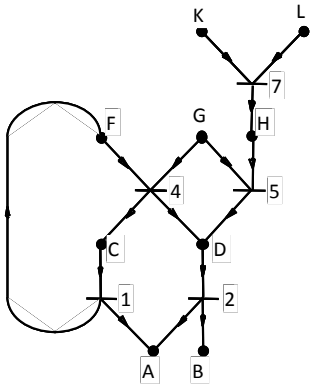
Solution #11



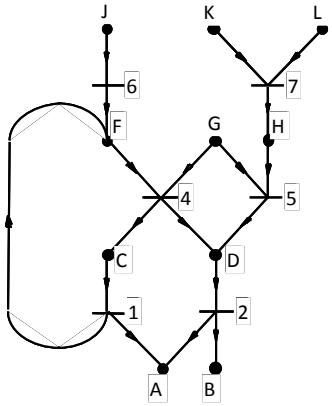
Solution #12



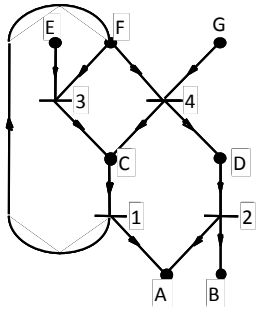
Solution #13



Solution #14

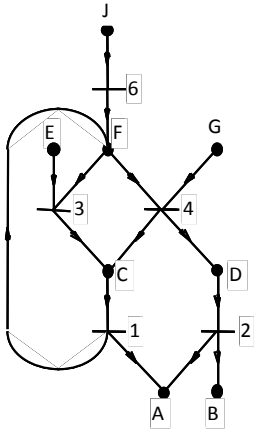


Solution #15

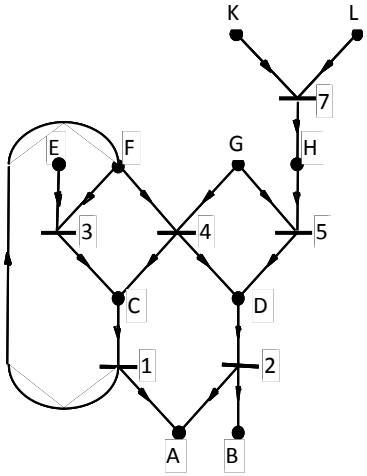


Solution #16

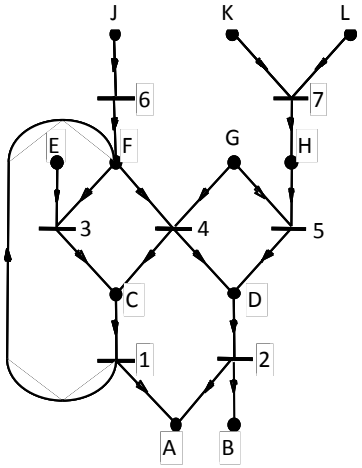
COMBINATORIALLY FEASIBLE STRUCTURES OF EXAMPLE PNS 1 (Cont'd)



Solution #17



Solution #18



Solution #19

SYNTHESIS OF AN INDUSTRIAL PROCESS (EXAMPLE PNS 2)

Product: A61

Raw materials: A1, A2, A3, A4, A6, A7, A8, A11, A15, A17, A18, A19, A20, A23, A27, A28, A29, A30, A34, A43, A47, A49, A52, A54

- The five axioms reduce the

34 billion combinations of the operating units to

3,465 combinatorially feasible structures.

The optimal solution is included in the set of 3465 feasible structures.

ILLUSTRATION OF THE REDUCTION IN THE SEARCH SPACE

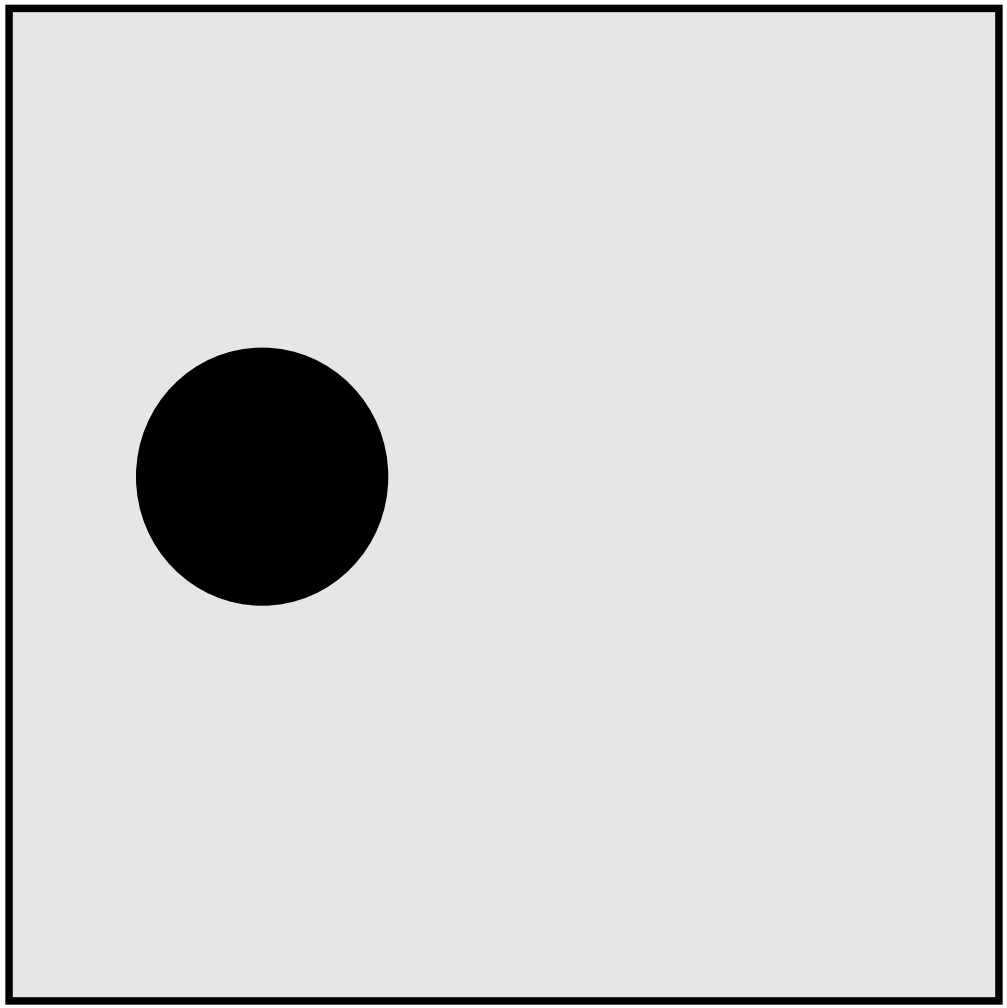
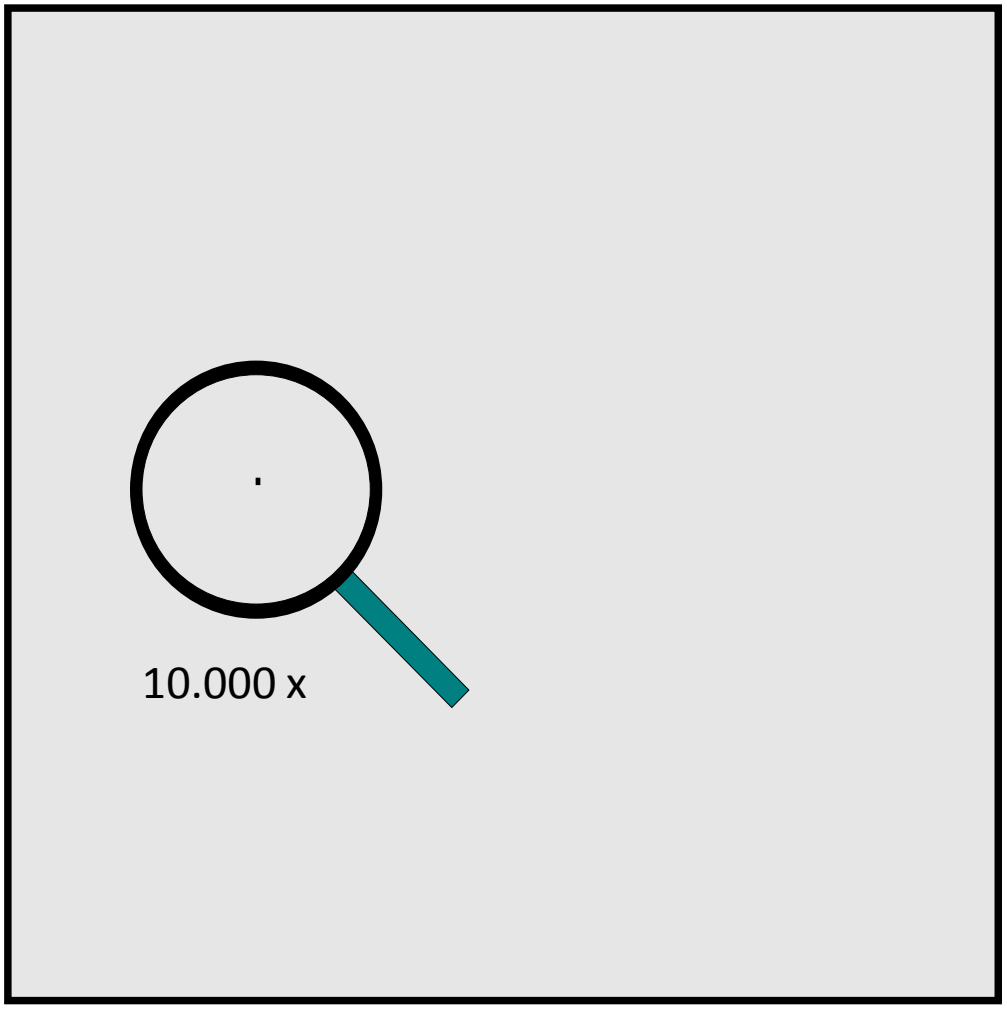


ILLUSTRATION OF THE REDUCTION IN THE SEARCH SPACE

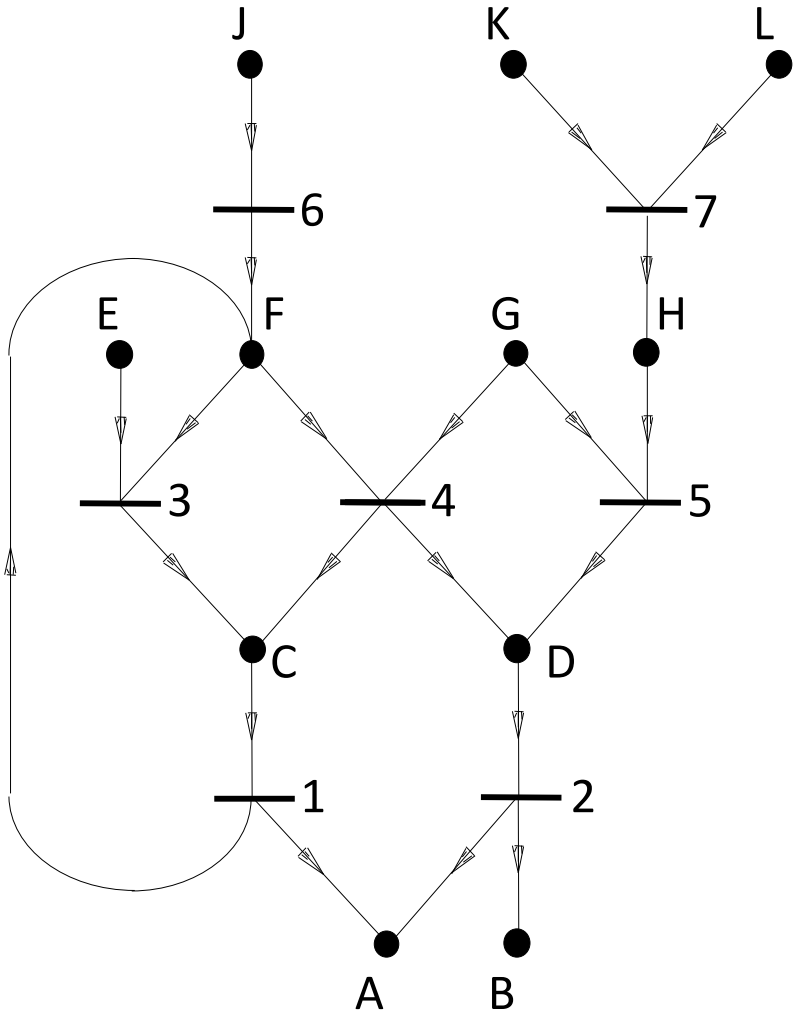


ALGORITHMIC GENERATION OF THE MAXIMAL STRUCTURE

MAXIMAL STRUCTURE

- The union of all combinatorially feasible structures is called the maximal structure.
- The maximal structure is a rigorous super-structure.

MAXIMAL STRUCTURE OF THE ILLUSTRATIVE EXAMPLE PNS 1



ALGORITHM MSG: GENERATION OF THE MAXIMAL STRUCTURE

- Input:

  Synthesis problem given by

  set of raw materials

  set of products

  set of candidate operating units

- Output:

  maximal structure

ALGORITHM MSG: GENERATION OF THE MAXIMAL STRUCTURE

inputs: sets M, P,R, O;

comment: $P \subseteq M$, $R \subseteq M$, $O \subseteq p(M) \times p(M)$, $O \cap M = \varnothing$, $P \cap R = \varnothing$;

output: maximal structure (m,o) of synthesis problem (P, R, O);

begin

reduction part of the algorithm;

composition part of the algorithm;

end

reduction part of the algorithm;

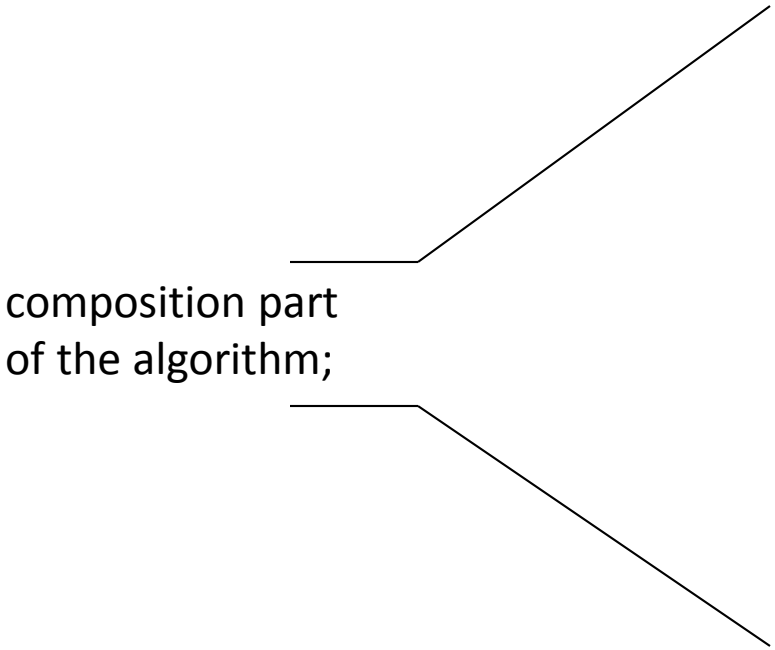
st1: $O:=O\setminus\varphi^{-}(R)$;

st2: $M:=\Psi(O)$;

st3: $r:=\Psi^{-}(O)\setminus(\Psi^{+}(O)\cup R)$;

lp4: **while** r is not empty **do**
begin
let x be an element of r;
$M:=M\setminus\{x\}$;
$o:=\varphi^{+}(\{x\})$;
$O:=O\setminus o$;
$r:=(r\cup(\Psi^{+}(o)\setminus\Psi^{+}(O)))\setminus\{x\}$;
end;

co5: **if** $P\cap M\neq P$ **then stop;**
comment: there is no maximal structure;



st6: p:=P; m:= ∅; o:= ∅;

lp7: **while** p is not empty **do**
begin
let x be an element of p;
m:=m∪{x};
$o_x := \varphi^-(\{x\})$;
o:=o∪ o_x;
p:=(p∪ $\Psi^-(o_x)$)\(R∪m);
end;

st8: m:= $\Psi(o)$;

Note: The complexity of algorithm MSG is polynomial.

Example (Generation of the maximal structure)

Materials:

M={A, B, C, D, E, F, G, H, I, J, K, L, M,N, Q, T, U, V},

Product:

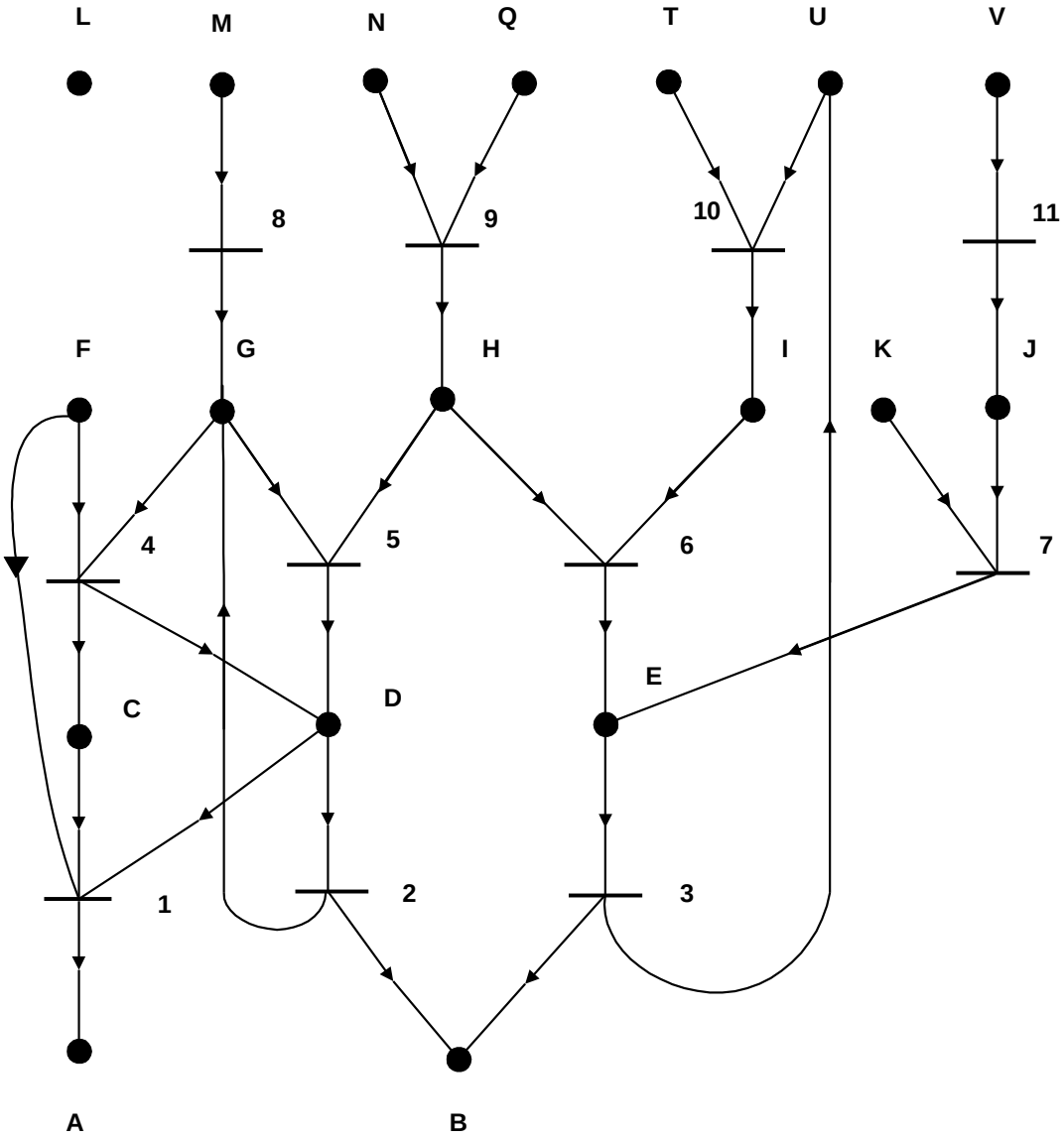
P={B}

Raw materials:

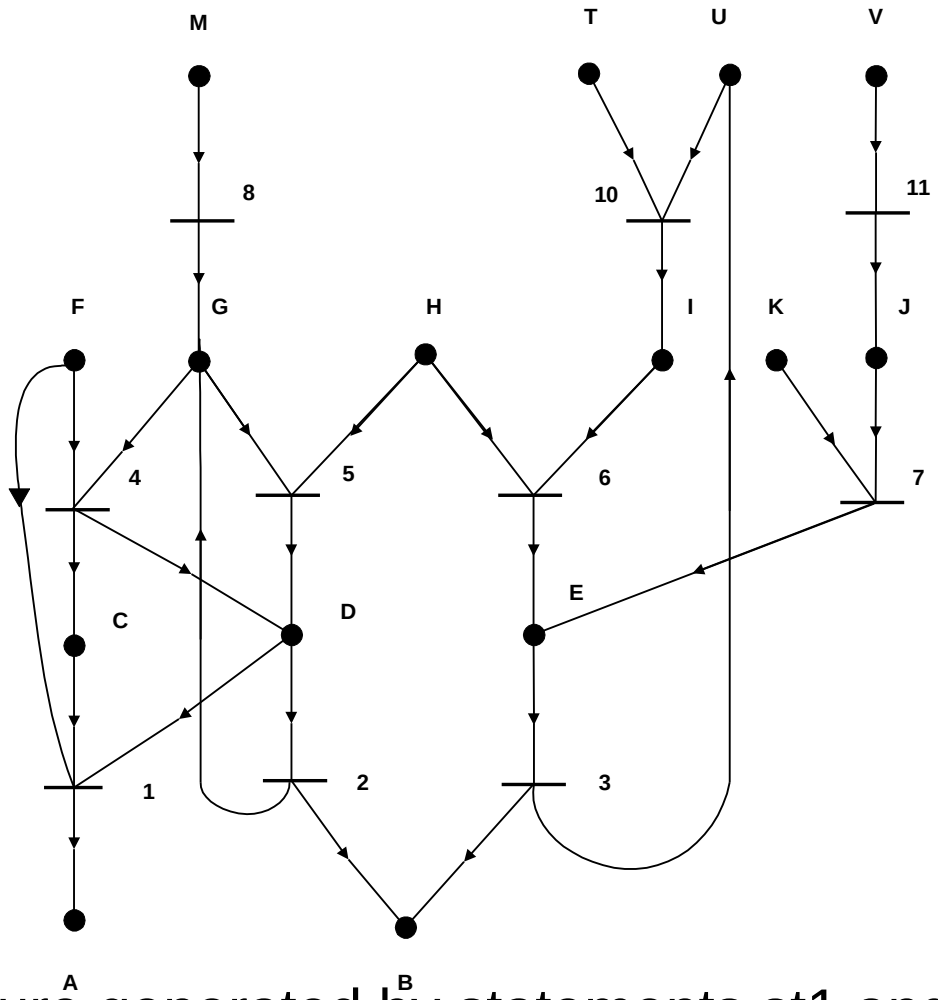
R={F, H, M, T}

Operating units:

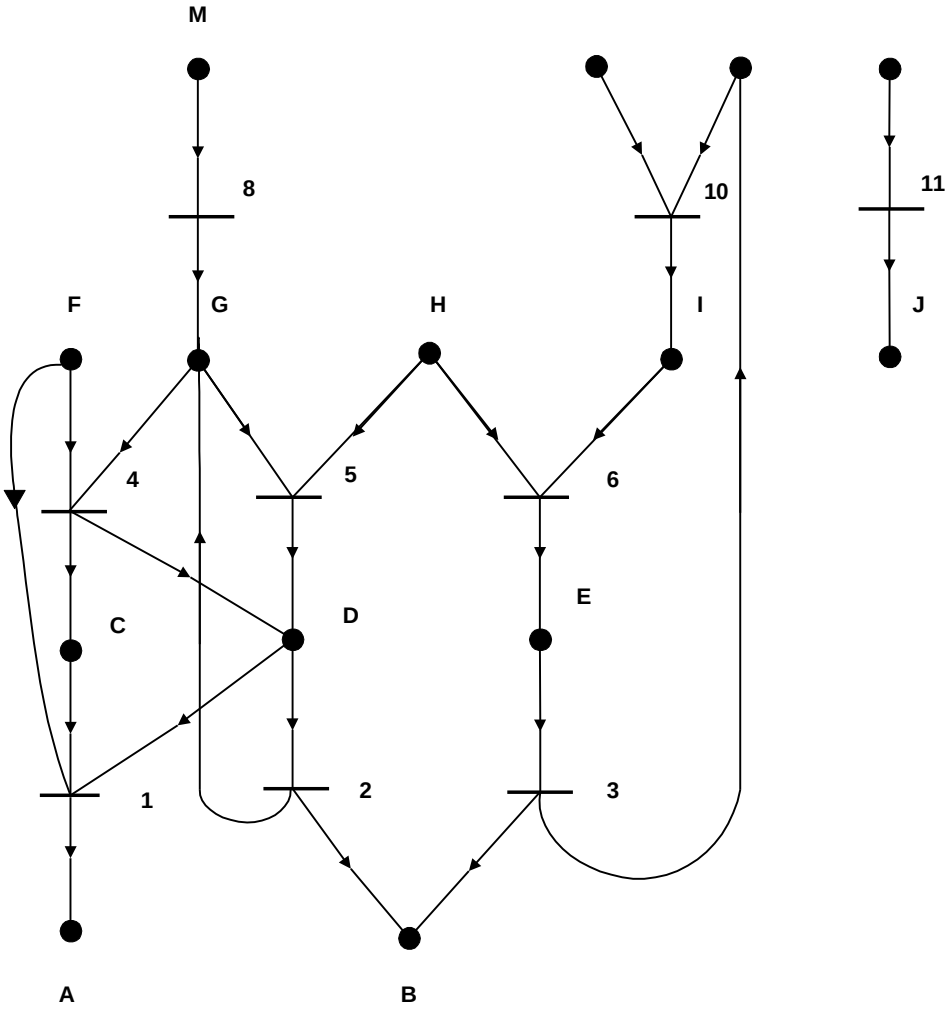
O={({C, D, F}, {A}), ({D}, {B,G}),
({E}, {B, U}), ({F, G}, {C, D}),
({G, H}, {D}), ({H, I}, {E}),
({J, K}, {E}), ({M}, {G}),
({N, Q}, {H}), ({T, U}, {I}),
({V}, {J})}.



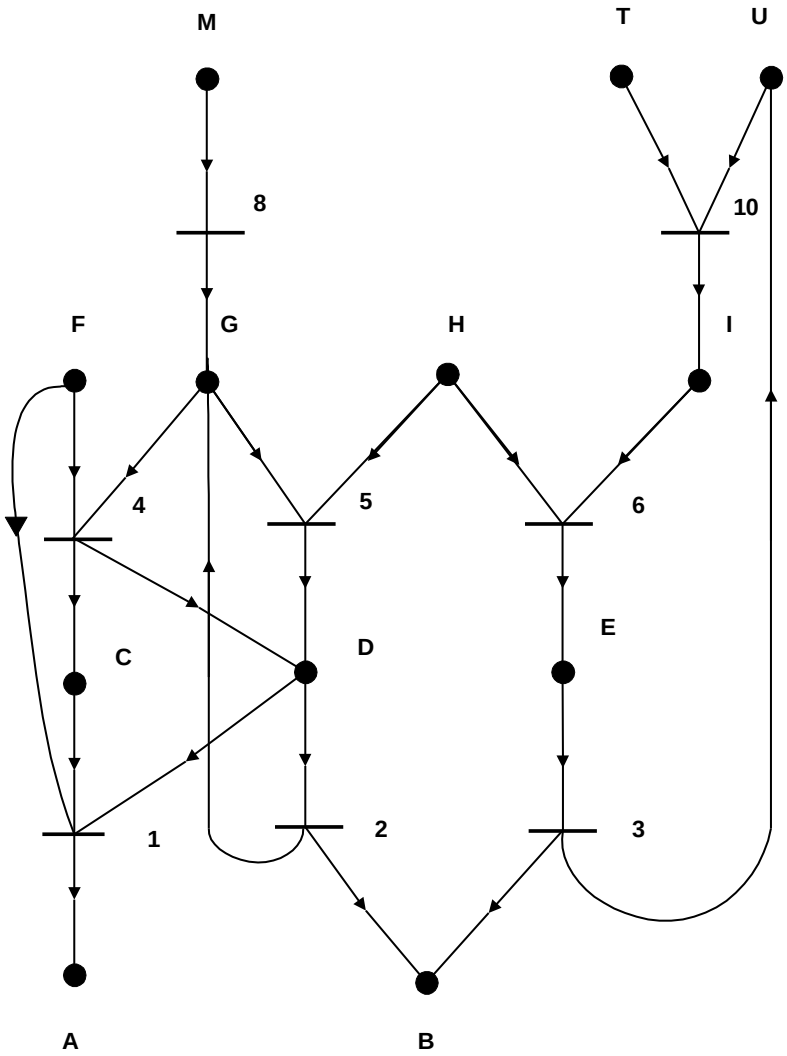
Input structure for algorithm MSG



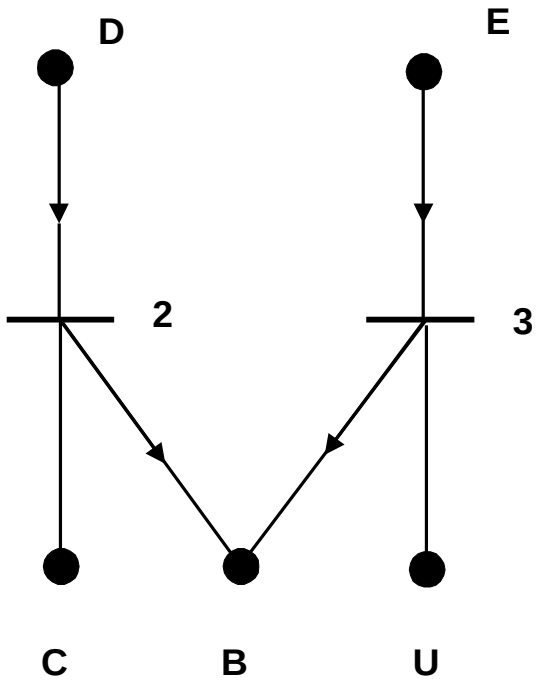
Sturcture generated by statements st1 and st2; materials belonging to set r of st3 are underlined.



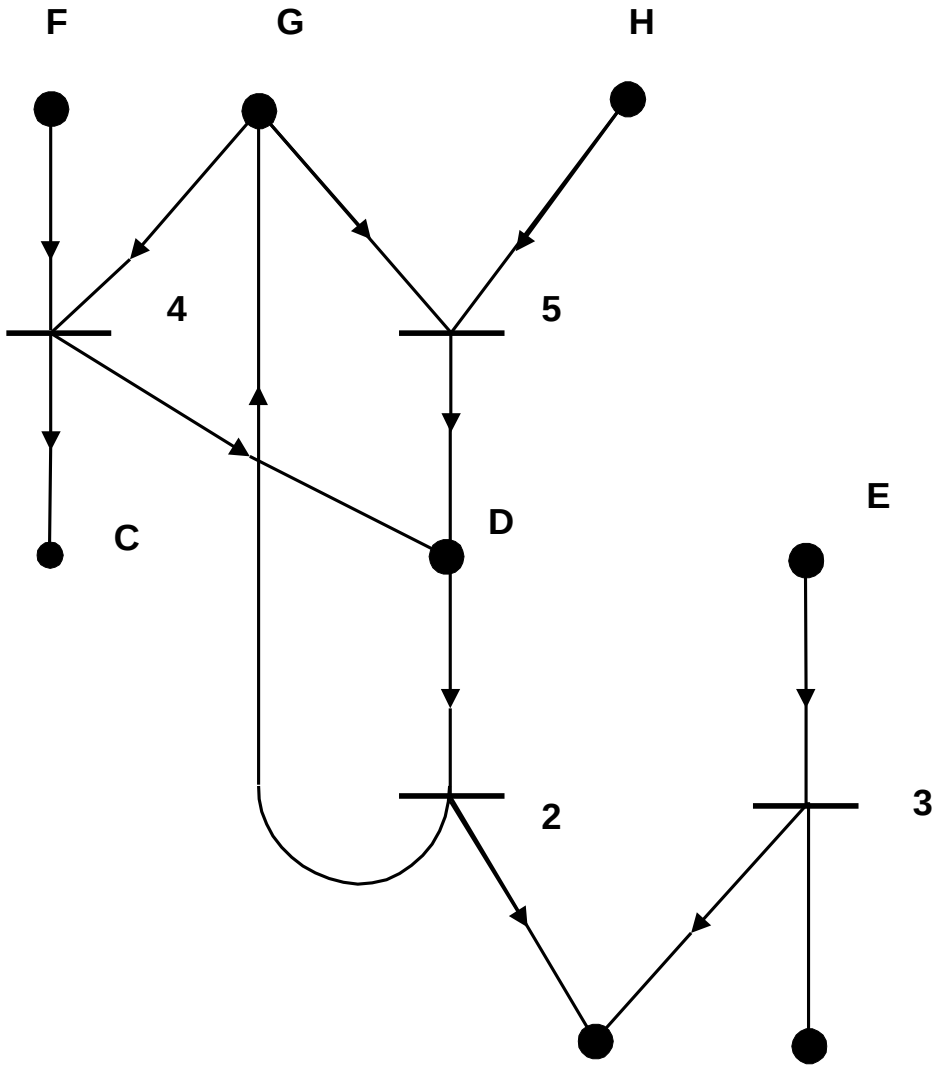
Structure generated after the first iteration of loop lp4.



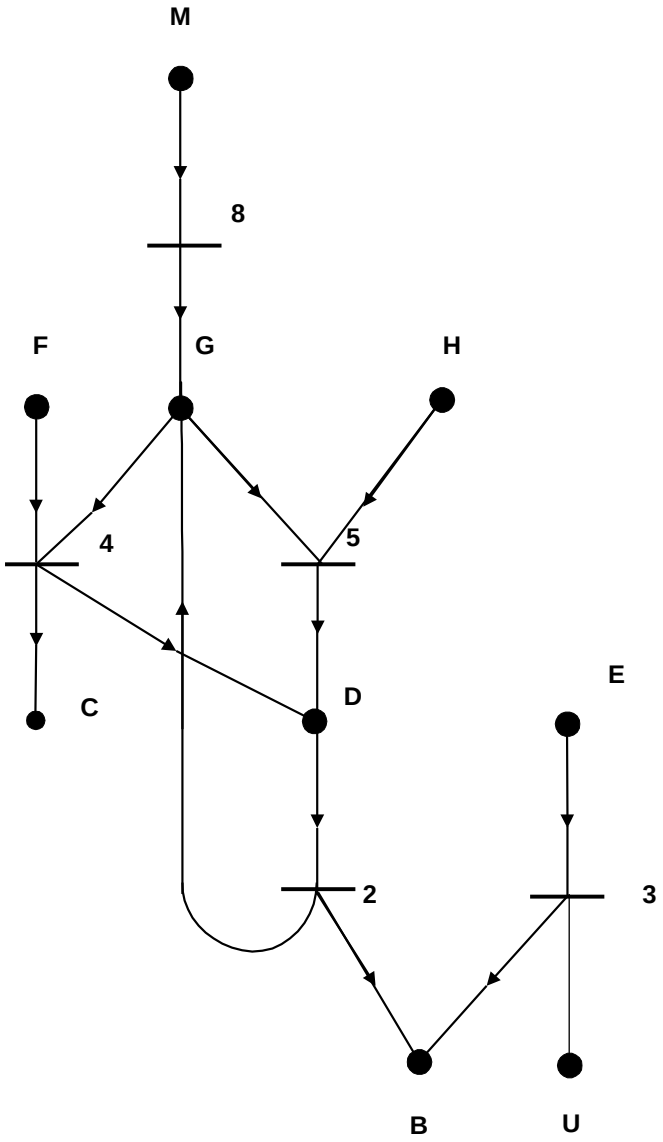
Structure generated after the third iteration of loop lp4.



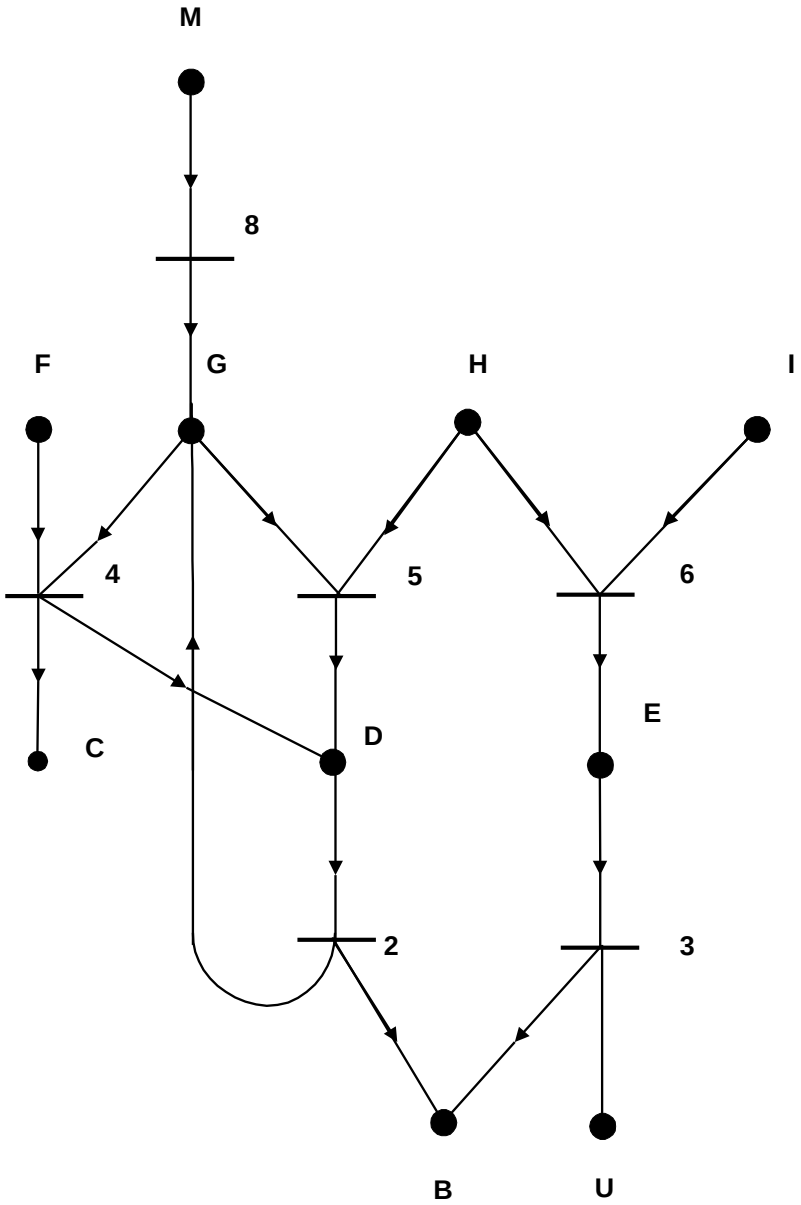
Structure of the first iteration of loop lp7.



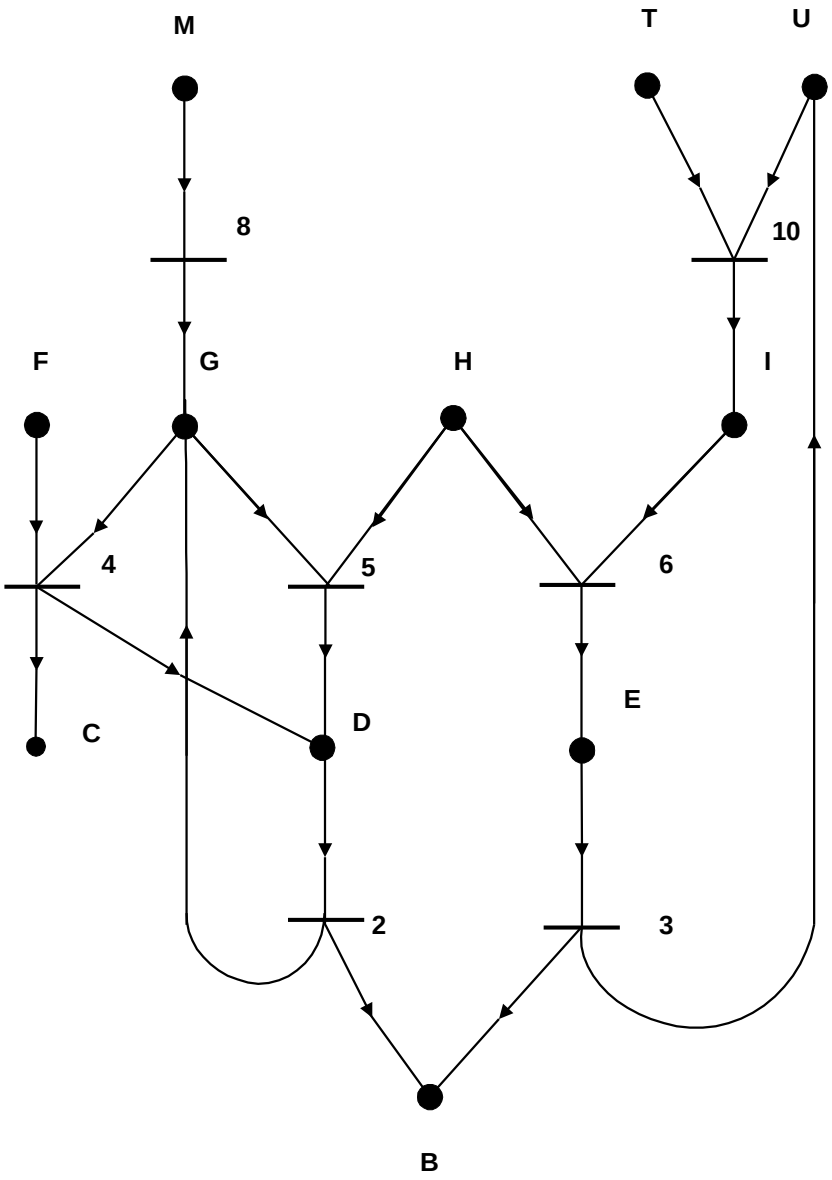
Structure of the second iteration of loop lp7.



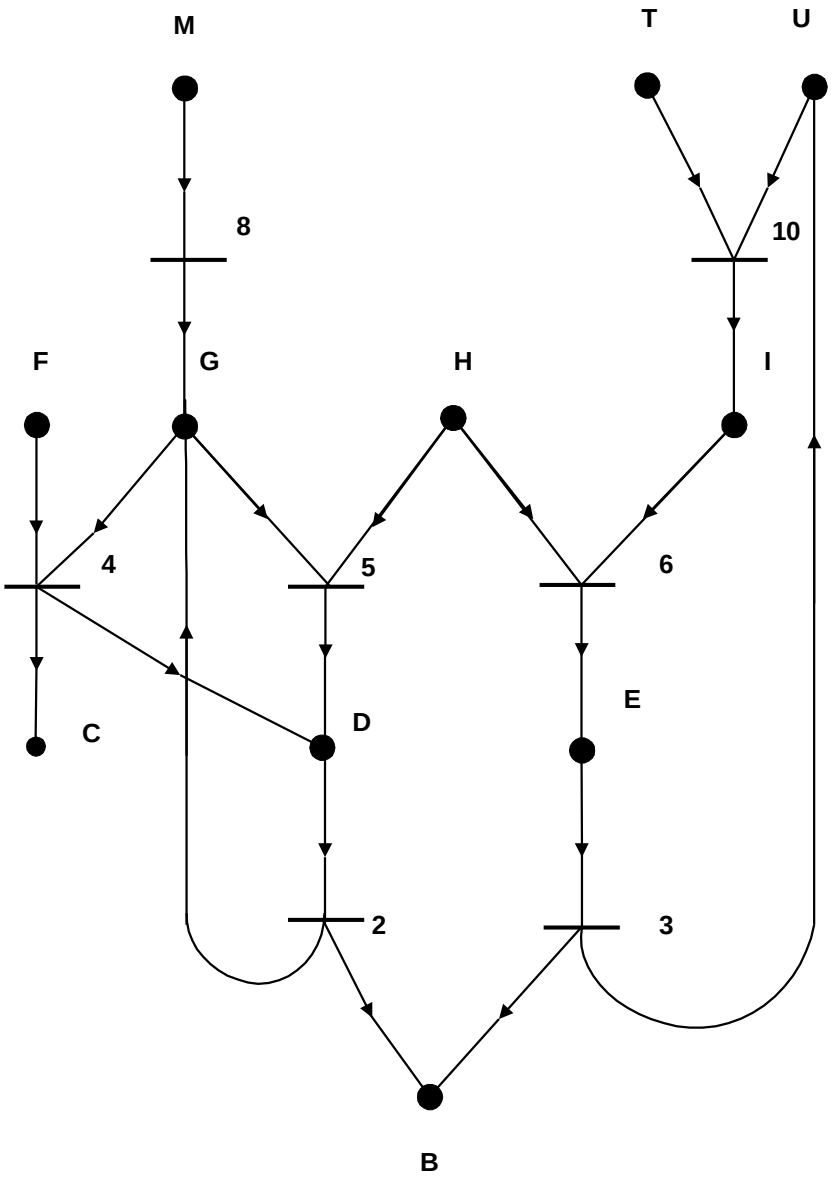
Structure after the third iteration of loop lp7.



Structure after the fourth iteration of loop lp7.



Structure after the fifth iteration of loop lp7.



Output of algorithm MSG: maximal structure

ALGORITHMIC GENERATION OF FEASIBLE STRUCTURES

OBSERVATION

The axioms are not in procedural form to generate process structures: additional tool is required

DECISION-MAPPING

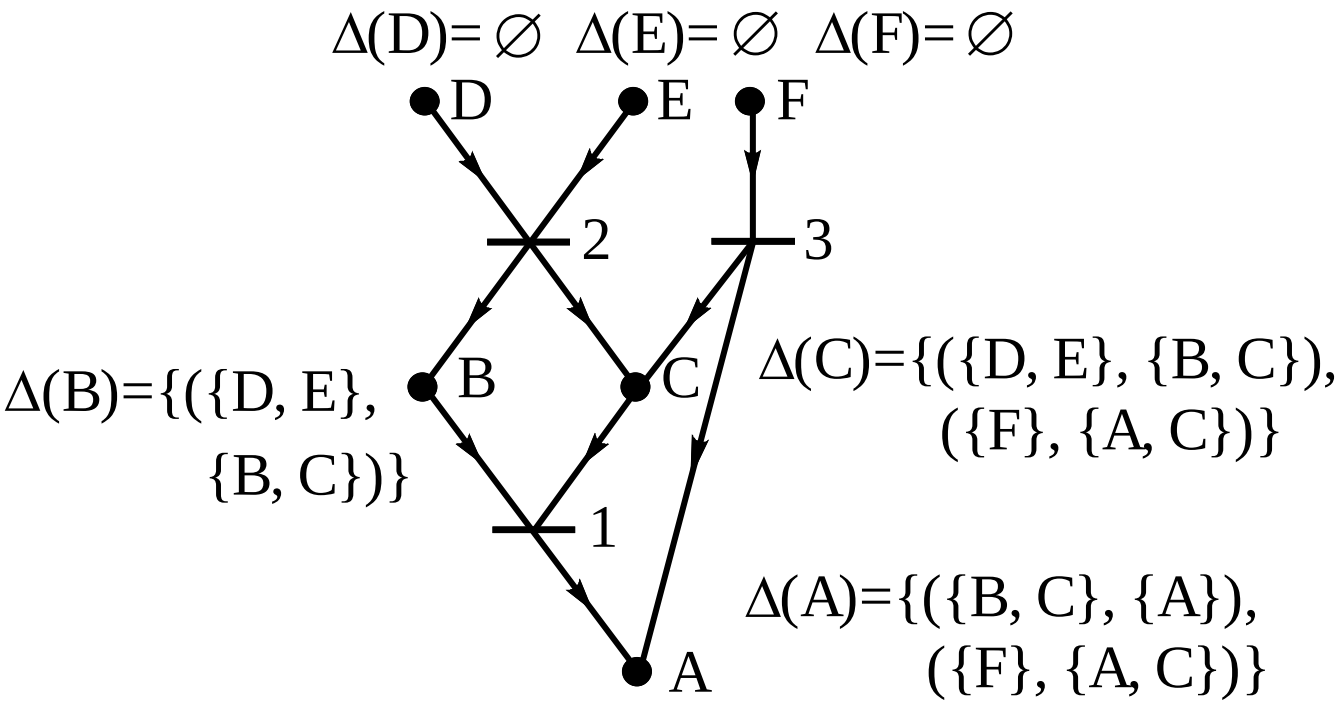
Decision-mapping is a novel mathematical notion to render the complex decisions in process synthesis consistent and complete.

FORMAL DEFINITION: DECISION MAPPING

- Mapping or function is a subset of a Cartesian product of domain D and range R.
- Function f is a set of pairs (x, y) where $x \in D$ and $y=f(x) \in R$; this set of pairs is denoted by f[D].
- Let Δ be a mapping from M to the set of subsets of O, i.e., $\Delta[M] \subseteq M \times \wp(O)$. This mapping determines the set of operating units producing material X for any $X \in M$.
- $\Delta(X)=\{(\alpha, \beta):(\alpha, \beta) \in O$ and $X \in \beta\}$ where m is a subset of M X is an element of m.
- $\delta[m]=\{(X, \delta(X)):X \in m\}$ is a decision mapping on m if $\delta(X)$ is a subset of $\Delta(X)$ for each $X \in m$.

(See Friedler *et al*., 1995b)

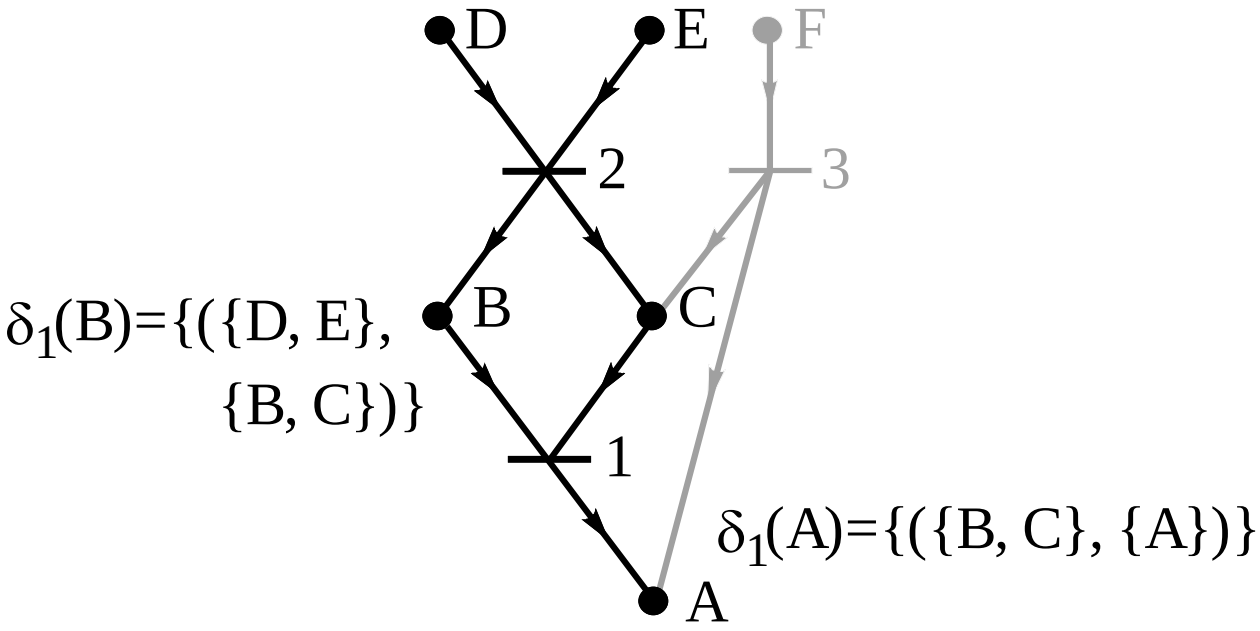
- Example



Maximal decision mapping Δ represents the whole structure where

Δ[{A, B, C, D, E,}]={(A, Δ(A)), (B, Δ(B)), (C, Δ(C)), (D, Δ(D)), (E, Δ(E)), (F, Δ(F))}

- Example (Cont'd)



Decision mapping $\delta 1$ represents a substructure where

$$\delta 1[\{A, B\}] = \{(A, \delta_1(A)), (B, \delta_1(B))\}$$

ALGORITHM SSG FOR GENERATING ALL SOLUTION-STRUCTURES OF A SYNTHESIS PROBLEM

- Input:
  Maximal structure
- Output:
  All solution-structures of the synthesis problem

ALGORITHM SSG FOR GENERATING ALL SOLUTION-STRUCTURES OF A SYNTHESIS PROBLEM

input: M, P, R, Δ[M];

comment: P, R, Δ[M] belong to synthesis problem (P, R, O), where

$P \subseteq M$, $R \subseteq M$, $P \cap R = \varnothing$, $\Delta(x) = \{(\alpha, \beta) | (\alpha, \beta) \in O \ \& \ x \in \beta\}$, $\Delta(x) = \varnothing \Leftrightarrow x \in R$,

$\Delta[M] = \{(x, \Delta(x)) | x \in M\}$, δ[m] is a decision-mapping on (M, O);

output: all solution-structures of synthesis problem (P, R, O);

global variables: R, Δ[M];

begin

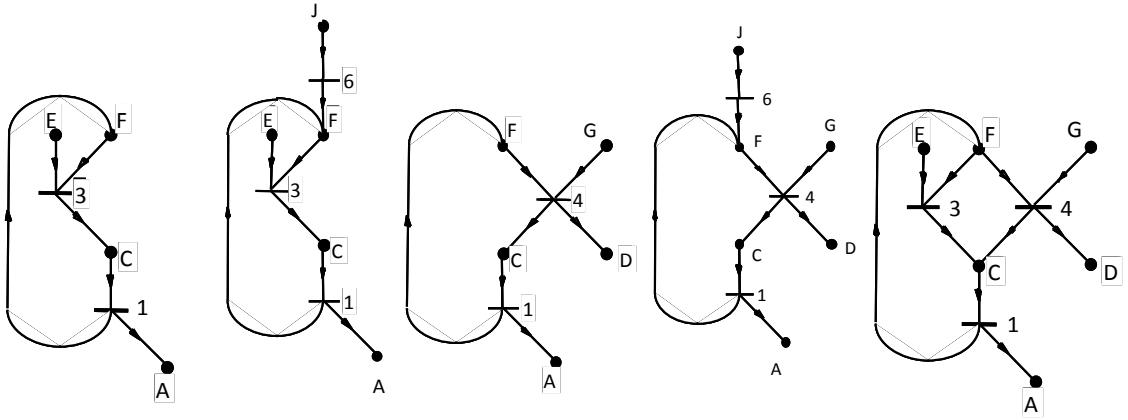
if $P = \varnothing$ **then stop**;

SSG(P, $\varnothing$, $\varnothing$)

end

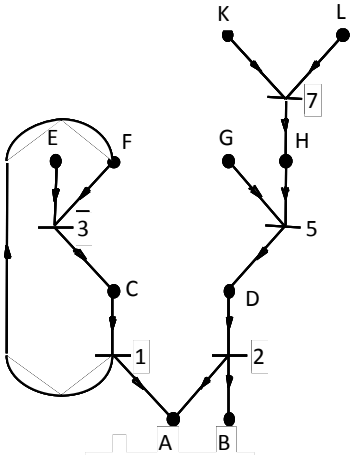
procedure SSG(p, m, δ[m])
begin
if p = $\varnothing$ **then begin write** δ[m]; **comment:** δ[m] defines a solution-structure;
return **end**
let x$\in$p;
C:= $\wp(\Delta(x))\backslash\{\varnothing\}$;
for all c$\in$C **do**
begin
if $\forall$y$\in$m, c$\cap\delta$(y) = $\varnothing$ & $(\Delta(x)\backslash c)\cap\delta(y) = \varnothing$
then
begin
$\delta[m\cup\{x\}]:= \delta[m]\cup\{(x, c)\}$;
SSG($p\cup mat^{in}$ (c))\(R$\cup$m$\cup${x}), m$\cup${x}, $\delta[m\cup\{x\}]$)
end
end
return
end

COMBINATORIALLY FEASIBLE STRUCTURES OF EXAMPLE PNS 1 GENERATED BY ALGORITHM SSG

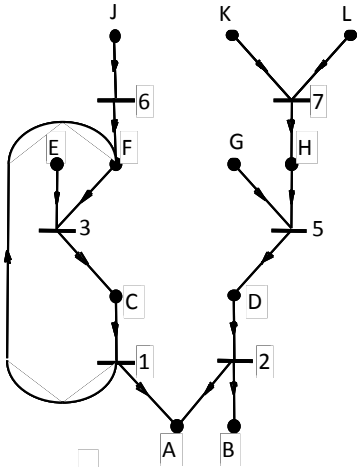


Solution #1 Solution #2 Solution #3 Solution #4 Solution #5

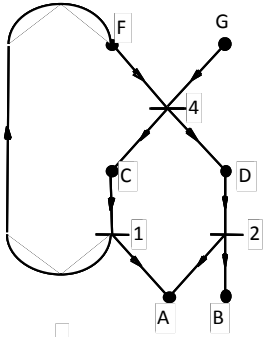
Solution #6 Solution #7 Solution #8 Solution #9



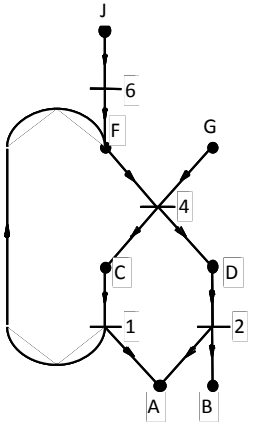
Solution #10



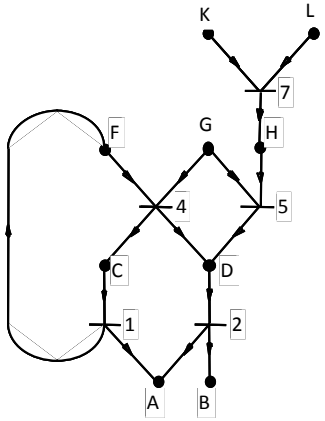
Solution #11



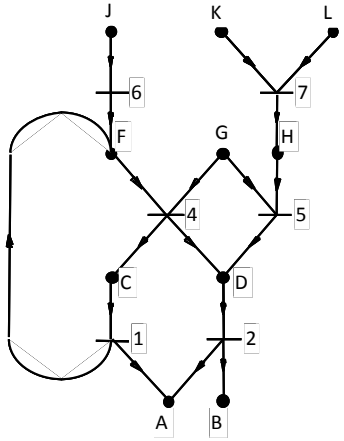
Solution #12



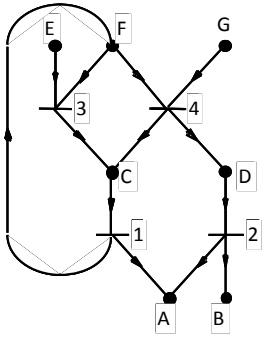
Solution #13



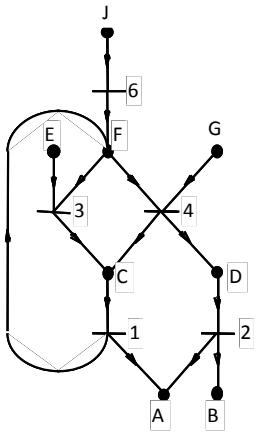
Solution #14



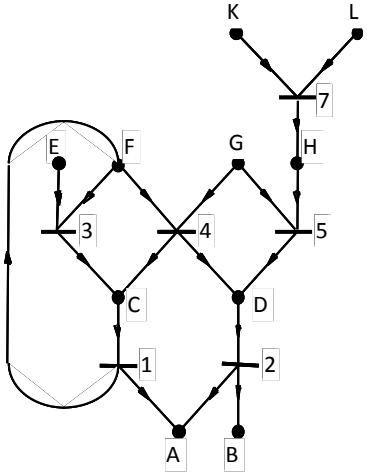
Solution #15



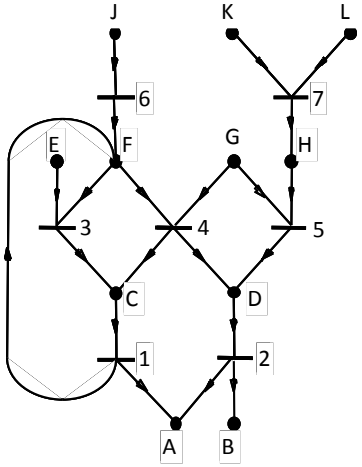
Solution #16



Solution #17



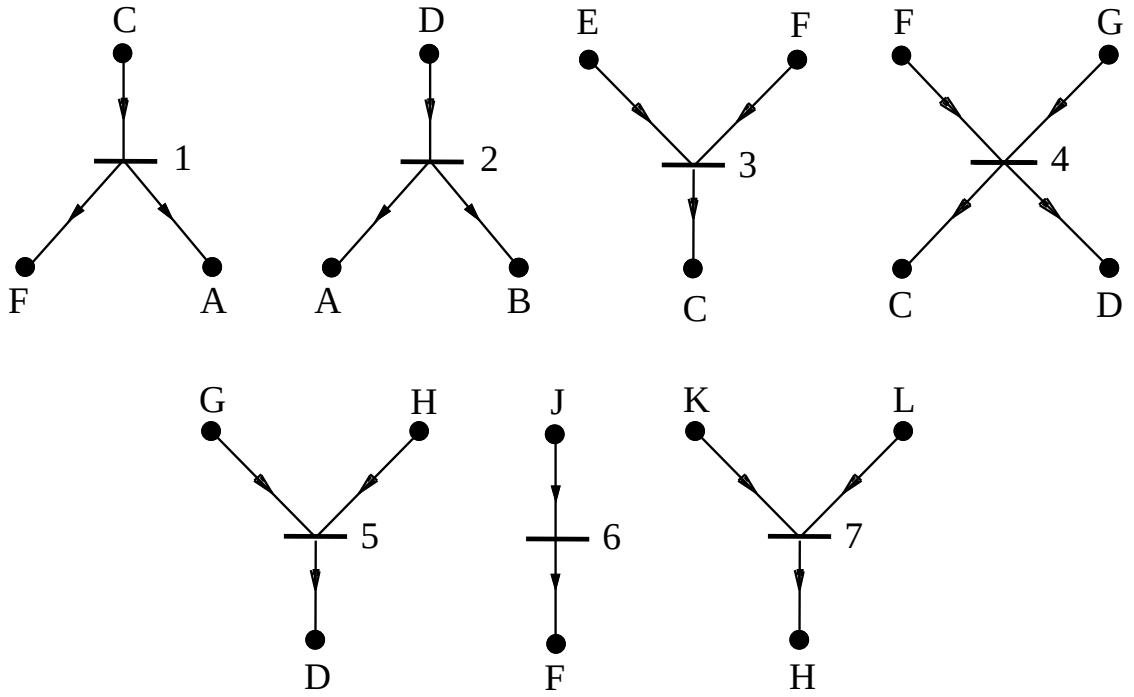
Solution #18



Solution #19

EXAMPLE PNS 1

- Operating units given



- Available raw materials: E, G, J, K, L
- Product: A

Depth of recursion: 0

{1}*, {2}, {1, 2}

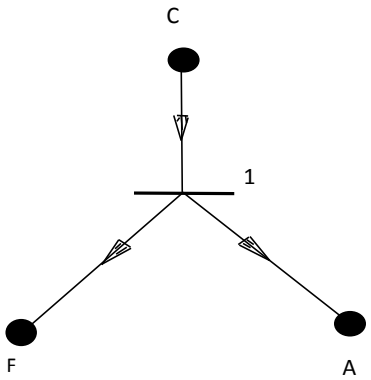
⇒ **procedure** SSG(p, m, δ[m]) p={A}, m=∅
begin
if p = ∅ **then begin write** δ[m]; p={A}
return **end**
let x∈p; p={A}, x=A;
C:= ℘(Δ(x))\{∅}; Δ(A)={1, 2}
for all c∈C **do** C={ {1}, {2}, {1, 2} }, c={1}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m=∅, c={1}, Δ(A)={1, 2}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m= ∅, x=A, c={1}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}]) δ[∅] = ∅
p={A}, mat^{in} ({1}) ={C}, R={E, J, G, K, L}
m=∅, x=A
end
end
return
end

Depth of recursion: 1

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

⇒ **procedure** SSG(p, m, δ[m]) p={C}, m={A}
begin
if p = ∅ **then begin write** δ[m]; p={C}
return **end**
let x∈p; p={C}, x=C;
C:= ℘(Δ(x))\{∅}; Δ(C)={3, 4}
for all c∈C **do** C={ {3}, {4}, {3, 4} }, c={3}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A}, c={3}, Δ(C)={3, 4}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A} , x=C, c={3}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={C}, mat^{in} ({3}) ={E, F}, R={E, J, G, K, L}
m={A}, x=C
end
end
return
end



δ[{A}]

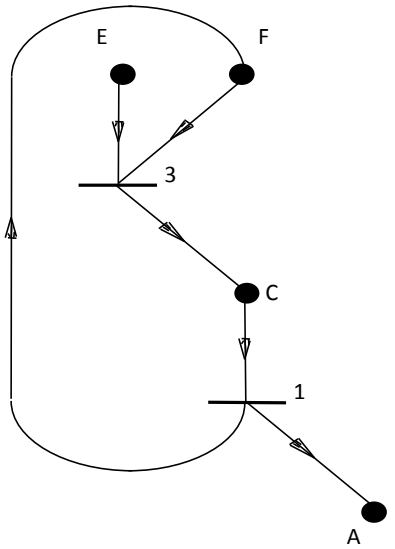
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}*, {6}, {1, 6}

⇒ **procedure** SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A, C}, c={1}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={F}, mat^{in} ({1}) ={C}, R={E, J, G, K, L}
m={A, C}, x=F
end
end
return
end



δ[{A, C}]

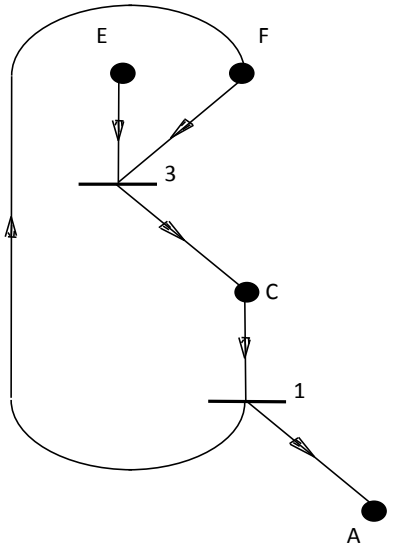
Depth of recursion: 3

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}*, {6}, {1, 6}

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= $\wp$(Δ(x))\{∅};
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end



δ[{A, C, F}]

Solution 1

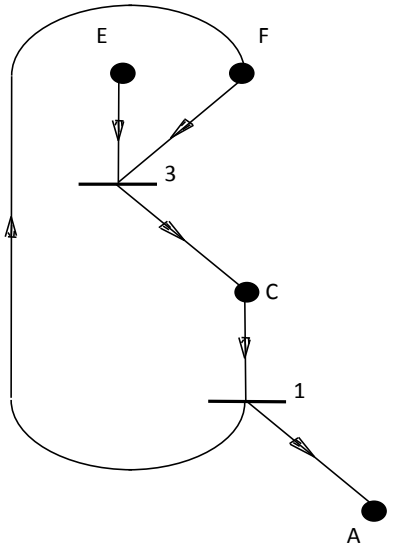
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}, **{6}***, {1, 6}

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return end
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ false
m={A, C}, c={6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
⇒ **end**
end
return
end



δ[{A, C}]

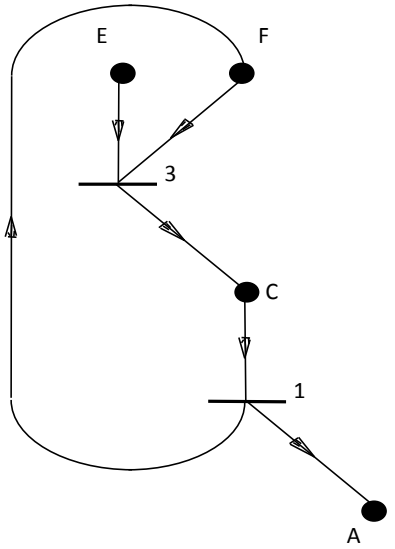
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1, 6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A, C}, c={1, 6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1, 6}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={F}, mat^{in} ({1, 6}) ={C, J}, R={E, J, G, K, L}
m={A, C}, x=F
end
end
return
end



δ[{A, C}]

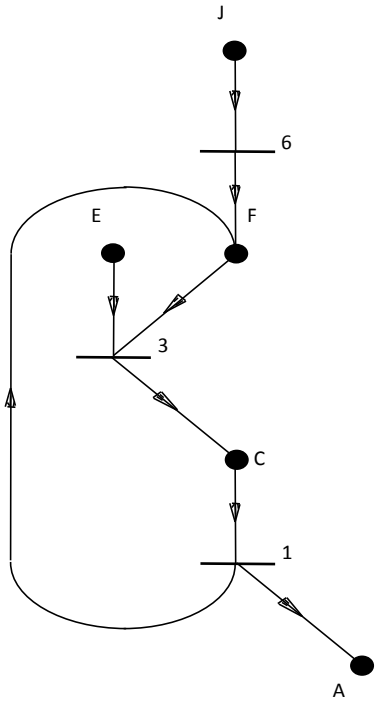
Depth of recursion: 3

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}, {6}, **{1, 6}***

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= $\wp$(Δ(x))\{∅};
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end



δ[{A, C, F}]

Solution 2

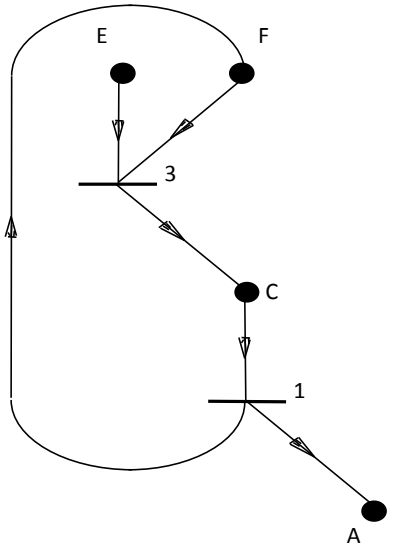
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}*, {4}, {3, 4}

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
⇒ **end**
⇐ **return**
end



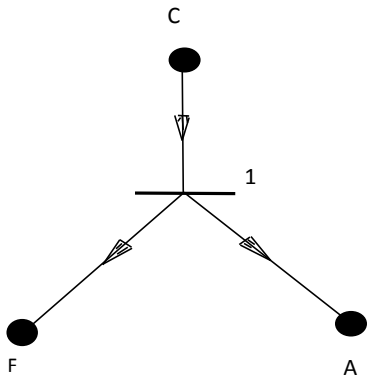
δ[{A, C}]

Depth of recursion: 1

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

procedure SSG(p, m, δ[m]) p={C}, m={A}
begin
if p = ∅ **then begin write** δ[m]; p={C}
return **end**
let x∈p; p={C}, x=C;
C:= ℘(Δ(x))\{∅}; Δ(C)={3, 4}
for all c∈C **do** C={ {3}, {4}, {3, 4} }, c={4}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A}, c={4}, Δ(C)={3, 4}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A} , x=C, c={4}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={C}, mat^{in} ({4}) ={F, G}, R={E, J, G, K, L}
m={A}, x=C
⇒ **end**
end
return
end



δ[{A}]

Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}*, {6}, {1, 6}

⇒ **procedure** SSG(p, m, δ[m]) p={F}, m={A, C}

begin

if p = ∅ **then begin write** δ[m]; p={F}

return **end**

let x∈p; p={F}, x=F;

C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}

for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1}

begin

if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true

m={A, C}, c={1}, Δ(F)={1, 6}

then

begin

δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1}

⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])

p={F}, mat^{in} ({1}) ={C}, R={E, J, G, K, L}

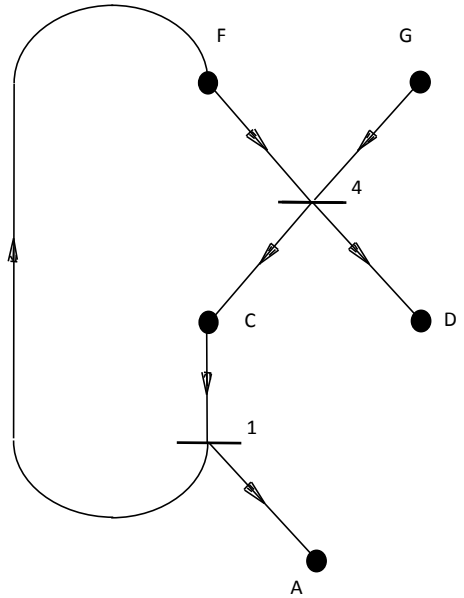
m={A, C}, x=F

end

end

return

end



δ[{A, C}]

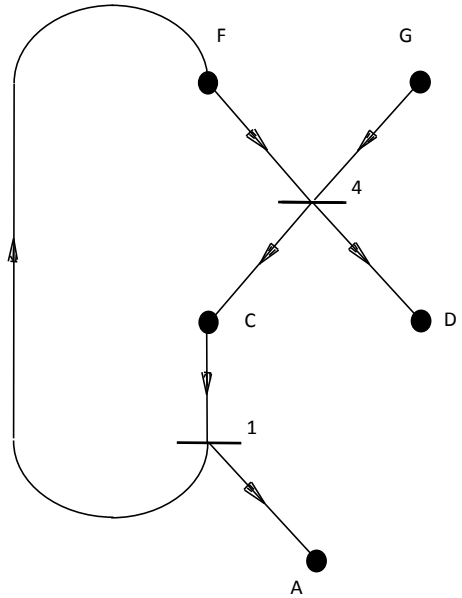
Depth of recursion: 3

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}*, {6}, {1, 6}

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= ℘(Δ(x))\{∅};
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end



δ[{A, C, F}]

Solution 3

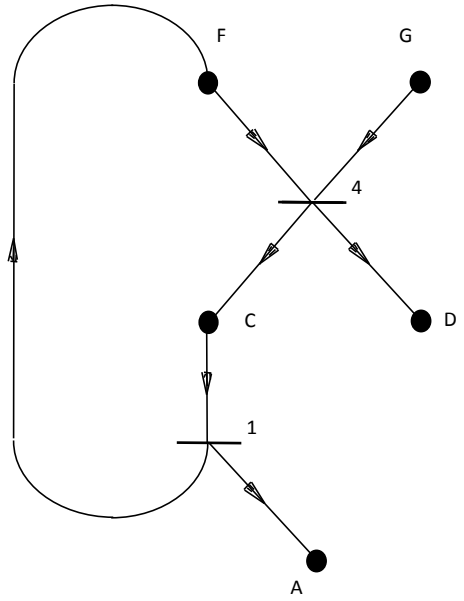
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}, **{6}***, {1, 6}

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return end
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ false
m={A, C}, c={6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
⇒ **end**
end
return
end



δ[{A, C}]

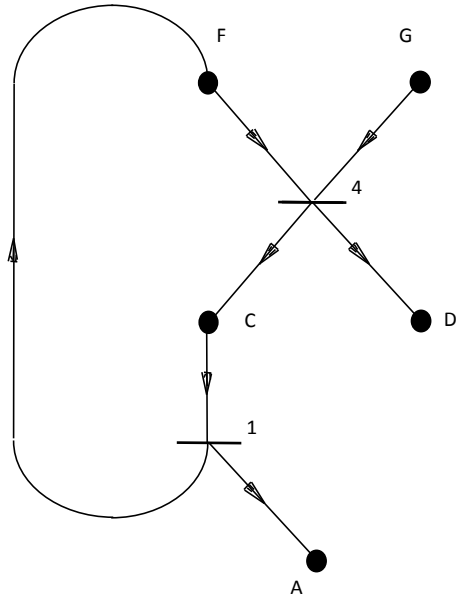
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1, 6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A, C}, c={1, 6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1, 6}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={F}, mat^{in} ({1, 6}) ={C, J}, R={E, J, G, K, L}
m={A, C}, x=F
end
end
return
end



δ[{A, C}]

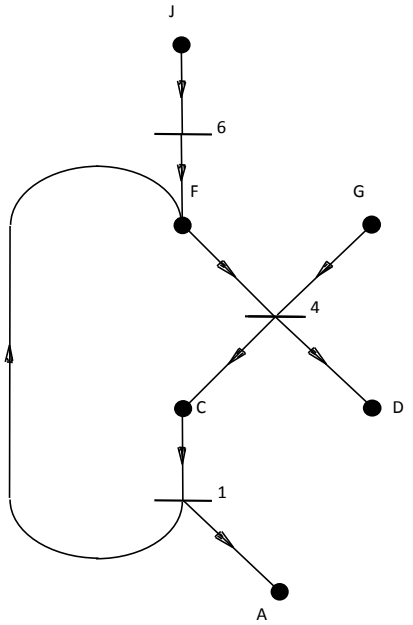
Depth of recursion: 3

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}, {6}, **{1, 6}***

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= $\wp(\Delta(x))\backslash\{\emptyset\}$;
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end



δ[{A, C, F}]

Solution 4

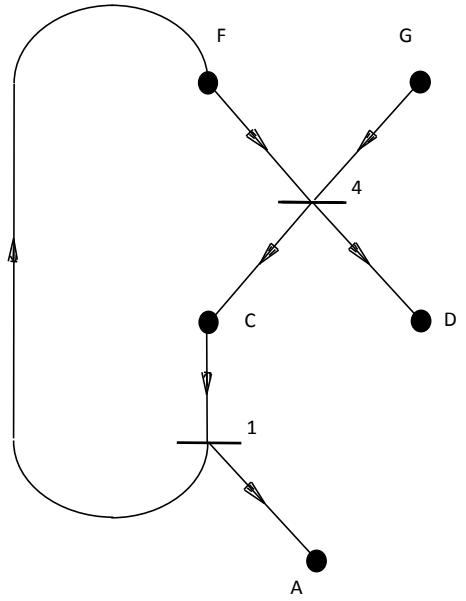
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, **{4}***, {3, 4}

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
⇒ **end**
⇐ **return**
end



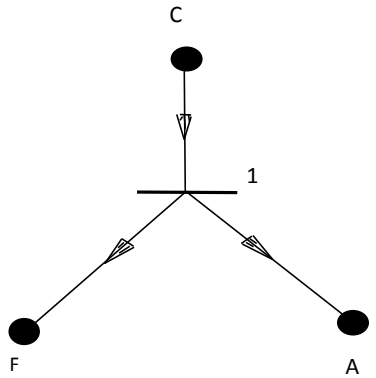
δ[{A, C}]

Depth of recursion: 1

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

procedure SSG(p, m, δ[m]) p={C}, m={A}
begin
if p = ∅ **then begin write** δ[m]; p={C}
return end
let x∈p; p={C}, x=C;
C:= ℘(Δ(x))\{∅}; Δ(C)={3, 4}
for all c∈C **do** C={ {3}, {4}, {3, 4} }, c={3, 4}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A}, c={3, 4}, Δ(C)={3, 4}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A} , x=C, c={3, 4}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={C}, mat^{in} ({3, 4}) ={E, F, G}, R={E, J, G, K, L}
m={A}, x=C
⇒ **end**
end
return
end



δ[{A}]

Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}*, {6}, {1, 6}

⇒ **procedure** SSG(p, m, δ[m]) p={F}, m={A, C}

begin

if p = ∅ **then begin write** δ[m]; p={F}

return end

let x∈p; p={F}, x=F;

C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}

for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1}

begin

if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true

m={A, C}, c={1}, Δ(F)={1, 6}

then

begin

δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1}

⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])

p={F}, mat^{in} ({1}) ={C}, R={E, J, G, K, L}

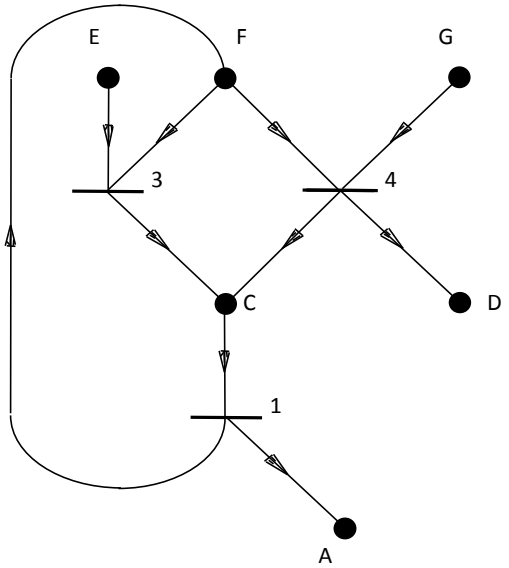
m={A, C}, x=F

end

end

return

end



δ[{A, C}]

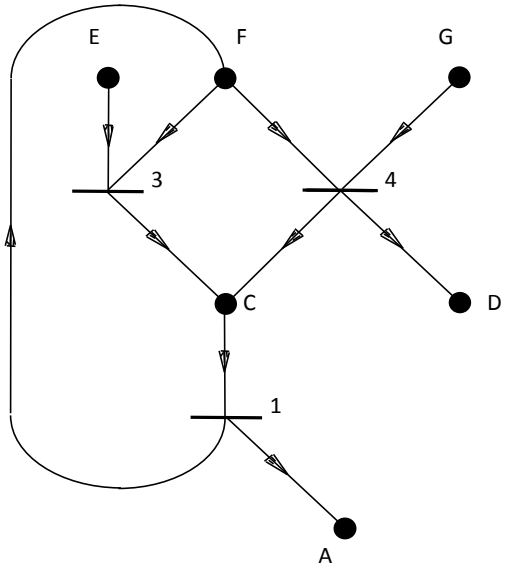
Depth of recursion: 3

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= ℘(Δ(x))\{∅};
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}*, {6}, {1, 6}



δ[{A, C, F}]

Solution 5

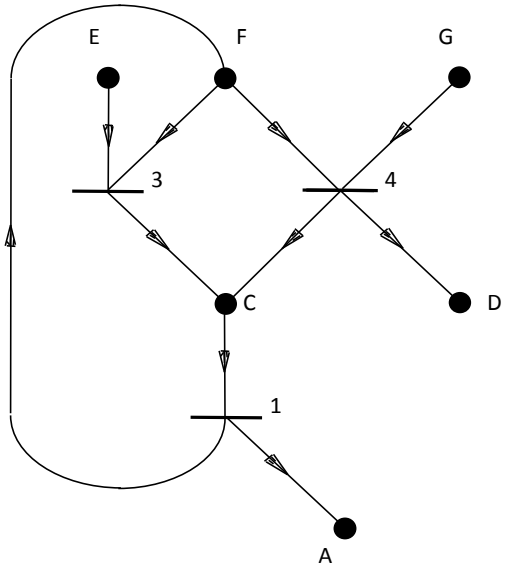
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}, **{6}***, {1, 6}

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return end
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ false
m={A, C}, c={6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
⇒ **end**
end
return
end



δ[{A, C}]

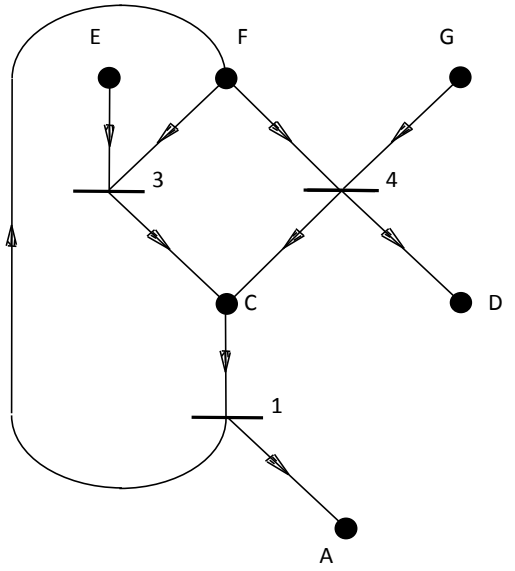
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }, c={1, 6}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m={A, C}, c={1, 6}, Δ(F)={1, 6}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m={A, C} , x=F, c={1, 6}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={F}, mat^{in} ({1, 6}) ={C, J}, R={E, J, G, K, L}
m={A, C}, x=F
end
end
return
end



δ[{A, C}]

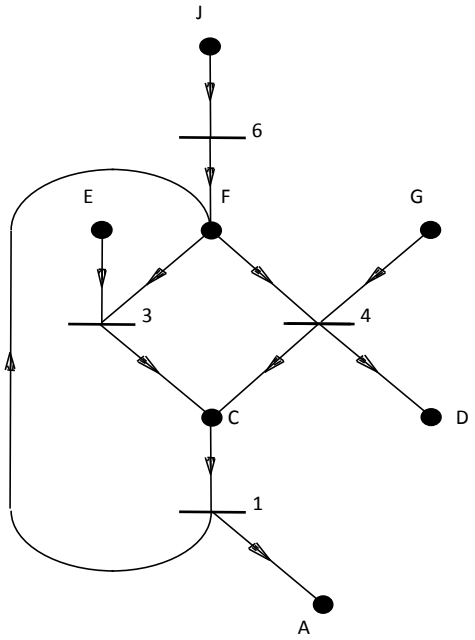
Depth of recursion: 3

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}, {6}, **{1, 6}***

⇒ **procedure** SSG(p, m, δ[m]) p=∅, m={A, C, F}
begin
if p = ∅ **then begin write** δ[m]; p=∅
⇐ **return end**
let x∈p;
C:= $\wp(\Delta(x))\backslash\{\emptyset\}$;
for all c∈C **do**
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
return
end



δ[{A, C, F}]

Solution 6

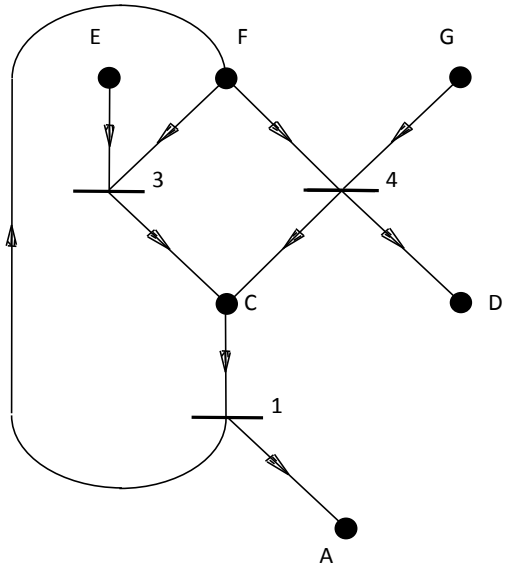
Depth of recursion: 2

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

{1}, {6}, **{1, 6}***

procedure SSG(p, m, δ[m]) p={F}, m={A, C}
begin
if p = ∅ **then begin write** δ[m]; p={F}
return **end**
let x∈p; p={F}, x=F;
C:= ℘(Δ(x))\{∅}; Δ(F)={1, 6}
for all c∈C **do** C={ {1}, {6}, {1, 6} }
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
⇒ **end**
end
⇐ **return**
end



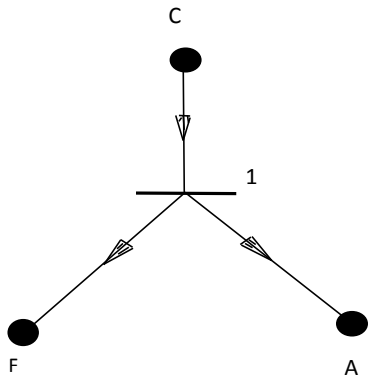
δ[{A, C}]

Depth of recursion: 1

{1}*, {2}, {1, 2}

{3}, {4}, **{3, 4}***

procedure SSG(p, m, δ[m]) p={C}, m={A}
begin
if p = ∅ **then begin write** δ[m]; p={C}
return **end**
let x∈p; p={C}, x=C;
C:= ℘(Δ(x))\{∅}; Δ(C)={3, 4}
for all c∈C **do** C={ {3}, {4}, {3, 4} }
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)};
SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
end
end
⇒
⇐ **return**
end



δ[{A}]

Depth of recursion: 0

{1}, **{2}***, {1, 2}

procedure SSG(p, m, δ[m]) p={A}, m=∅
begin
if p = ∅ **then begin write** δ[m]; p={A}
return **end**
let x∈p; p={A}, x=A;
C:= ℘(Δ(x))\{∅}; Δ(A)={1, 2}
for all c∈C **do** C={ {1}, {2}, {1, 2} }, c={2}
begin
if ∀y∈m, c∩δ(y) = ∅ & (Δ(x)\c)∩δ(y) = ∅ true
m=∅, c={2}, Δ(A)={1, 2}
then
begin
δ[m∪{x}]:= δ[m]∪{(x, c)}; m= ∅, x=A, c={2}
⇐ SSG(p∪mat^{in} (c))\(R∪m∪{x}), m∪{x}, δ[m∪{x}])
p={A}, mat^{in} ({2}) ={D}, R={E, J, G, K, L}
m=∅, x=A
⇒ **end**
end
return
end

δ[∅] = ∅

RECURSIVE STEPS OF ALGORITHM SSG

Number of call	Depth of recursion	Parameter p	Parameter m	Parameter δ[m]	Remark
1	0	{A}	∅	∅	Initial call
2	1	{C}	{A}	{(A,{1})}	
3	2	{F}	{A,C}	{(A,{1}),(C,{3})}	
4	3	∅	{A,C,F}	{(A,{1}),(C,{3}),(F,{1})}	Solution #1
5	3	∅	{A,C,F}	{(A,{1}),(C,{3}),(F,{1,6})}	Solution #2
6	2	{F}	{A,C}	{(A,{1}),(C,{4})}	
7	3	∅	{A,C,F}	{(A,{1}),(C,{4}),(F,{1})}	Solution #3
8	3	∅	{A,C,F}	{(A,{1}),(C,{4}),(F,{1,6})}	Solution #4
9	2	{F}	{A,C}	{(A,{1}),(C,{3,4})}	
10	3	∅	{A,C,F}	{(A,{1}),(C,{3,4}),(F,{1})}	Solution #5
11	3	∅	{A,C,F}	{(A,{1}),(C,{3,4}),(F,{1,6})}	Solution #6
12	1	{D}	{A}	{(A,{2})}	
13	2	{F}	{A,D}	{(A,{2}),(D,{4})}	
14	3	∅	{A,D,F}	{(A,{2}),(D,{4}),(F,{6})}	Solution #7

RECURSIVE STEPS OF ALGORITHM SSG (Cont'd)

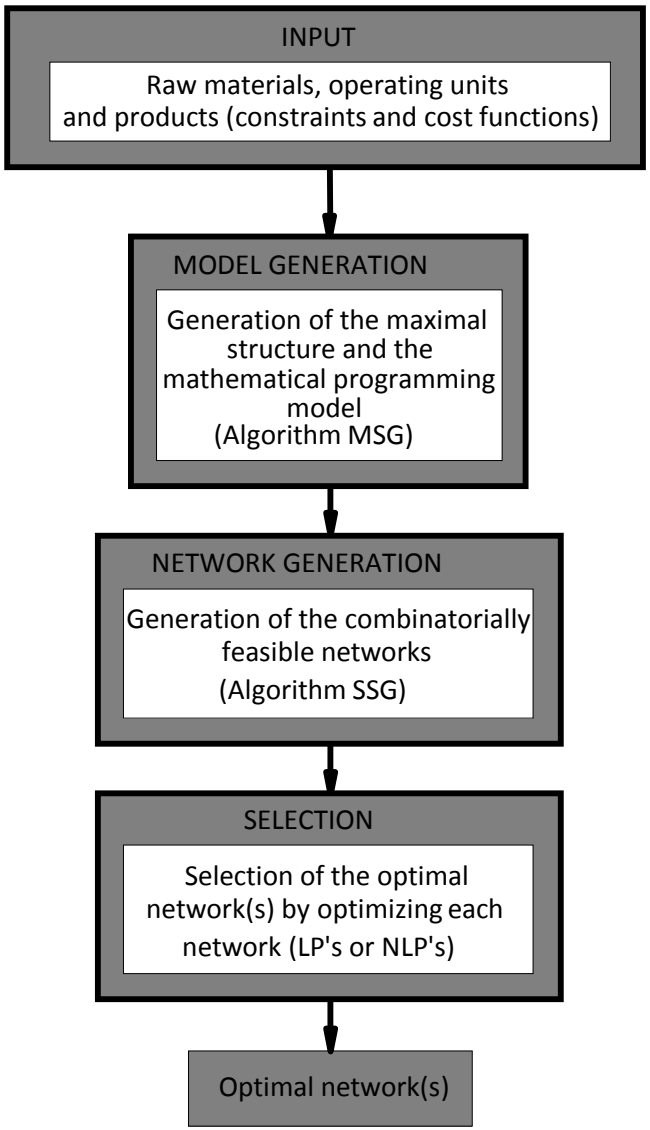
Number of call	Depth of recursion	Parameter p	Parameter m	Parameter δ[m]	Remark
15	2	{H}	{A,D}	{(A,{2}),(D,{5})}	
16	3	∅	{A,D,H}	{(A,{2}),(D,{5}),(H,{7})}	Solution #8
17	2	{F,H}	{A,D}	{(A,{2}),(D,{4,5})}	
18	3	{H}	{A,D,F}	{(A,{2}),(D,{4,5}),(F,{6})}	
19	4	∅	{A,D,F,H}	{(A,{2}),(D,{4,5}),(F,{6}),(H,{7})}	Solution #9
20	1	{C,D}	{A}	{(A,{1,2})}	
21	2	{D,F}	{A,C}	{(A,{1,2}),(C,{3})}	
22	3	{F,H}	{A,C,D}	{(A,{1,2}),(C,{3}),(D,{5})}	
23	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{3}),(D,{5}),(F,{1})}	
24	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{3}),(D,{5}),(F,{1}),(H,{7})	Solution #10
25	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{3}),(D,{5}),(F,{1,6})}	
26	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{3}),(D,{5}),(F,{1,6}),(H,{7})}	Solution #11
27	2	{D,F}	{A,C}	{(A,{1,2}),(C,{4})}	
28	3	{F}	{A,C,D}	{(A,{1,2}),(C,{4}),(D,{4})}	
29	4	∅	{A,C,D,F}	{(A,{1,2}),(C,{4}),(D,{4}),(F,{1})}	Solution #12
30	4	∅	{A,C,D,F}	{(A,{1,2}),(C,{4}),(D,{4}),(F,{1,6})}	Solution #13
31	3	{F,H}	{A,C,D}	{(A,{1,2}),(C,{4}),(D,{4,5})}	

RECURSIVE STEPS OF ALGORITHM SSG (Cont'd)

Number of call	Depth of recursion	Parameter p	Parameter m	Parameter δ[m]	Remark
32	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{4}),(D,{4,5}),(F,{1})}	
33	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{4}),(D,{4,5}),(F,{1}),(H,{7})}	Solution #14
34	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{4}),(D,{4,5}),(F,{1,6})}	
35	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{4}),(D,{4,5}),(F,{1,6}),(H,{7})}	Solution #15
36	2	{D,F}	{A,C}	{(A,{1,2}),(C,{3,4})}	
37	3	{F}	{A,C,D}	{(A,{1,2}),(C,{3,4}),(D,{4})}	
38	4	∅	{A,C,D,F}	{(A,{1,2}),(C,{3,4}),(D,{4}),(F,{1})}	Solution #16
39	4	∅	{A,C,D,F}	{(A,{1,2}),(C,{3,4}),(D,{4}),(F,{1,6})}	Solution #17
40	3	{F,H}	{A,C,D}	{(A,{1,2}),(C,{3,4}),(D,{4,5})}	
41	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{3,4}),(D,{4,5}),(F,{1})}	
42	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{3,4}),(D,{4,5}),(F,{1}),(H,{7})}	Solution #18
43	4	{H}	{A,C,D,F}	{(A,{1,2}),(C,{3,4}),(D,{4,5}),(F,{1,6})}	
44	5	∅	{A,C,D,F,H}	{(A,{1,2}),(C,{3,4}),(D,{4,5}),(F,{1,6}),(H,{7})}	Solution #19

ALGORITHMIC SYNTHESIS BY EXHAUSTIVE SEARCH

ALGORITHMIC SYNTHESIS BY EXHAUSTIVE SEARCH



SELECTION OF THE OPTIMAL NETWORK(S)

- The combinatorial algorithms, MSG, SSG, are independent of the type of mathematical model of the operating units.
- The exhaustive search will be illustrated by two types of models.
- Case I.
  - Linear cost functions and models of the operating units (MILP).
  - Solution procedure: sequence of LP-s.

Note: Algorithm SSG transforms the MILP problem into a sequence of LP-s.

- Case II.
  - Nonlinear cost functions and linear models of the operating units (MINLP).
  - Solution procedure: sequence of SSG-NLP where the cost function of the NLP is separable concave.

Note: this class of NLP problems can be solved effectively, see, e.g., Falk and Soland, 1969

OBSERVATION

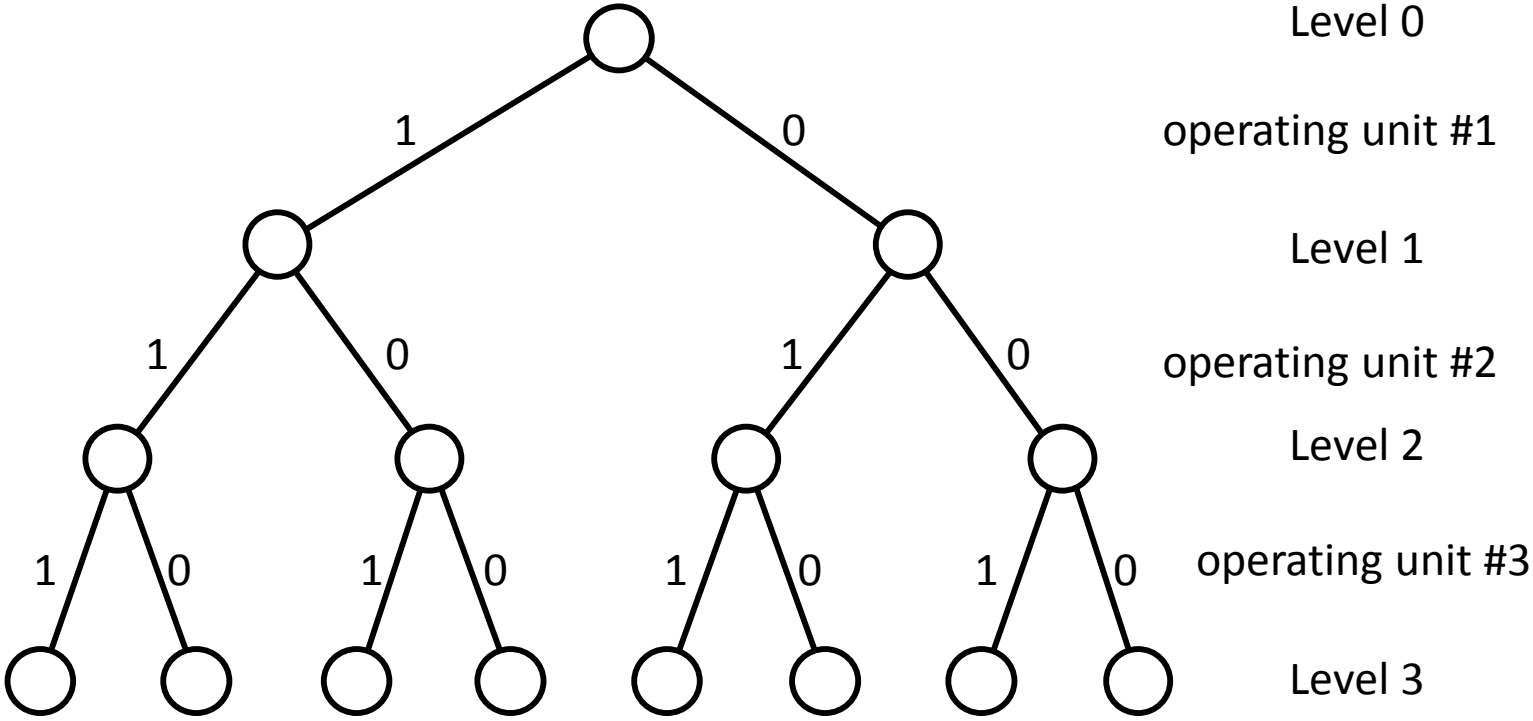
- The combinatorial axioms may drastically reduce the search space so that synthesis problems can be solved by exhaustive search (Algorithm SSG).
- The combinatorial part of the synthesis problem may effectively control the synthesis procedure if the search space can be reduced algorithmically.
- For very complex problems, this reduction may not be enough; further acceleration may be necessary.
- Possible way: branch-and-bound exploiting the reduced search space given by the axioms.

ACCELERATED BRANCH & BOUND ALGORITHM FOR SOLVING PNS PROBLEMS

ON THE BRANCH-AND-BOUND ALGORITHM

- Branch-and-bound search is a possible way for solving the MILP or MINLP problems.
- Branch-and-bound generates the optimal solution by solving a system of simplified LP or NLP partial problems by successively partitioning the solution set.
- Suppose that a binary variable expresses the existence or absence of an operating unit (the value is 1 for the former and 0 for the latter).

THE BASIC BRANCH-AND-BOUND SEARCH ILLUSTRATED ON AN ENUMERATION TREE



Notation:

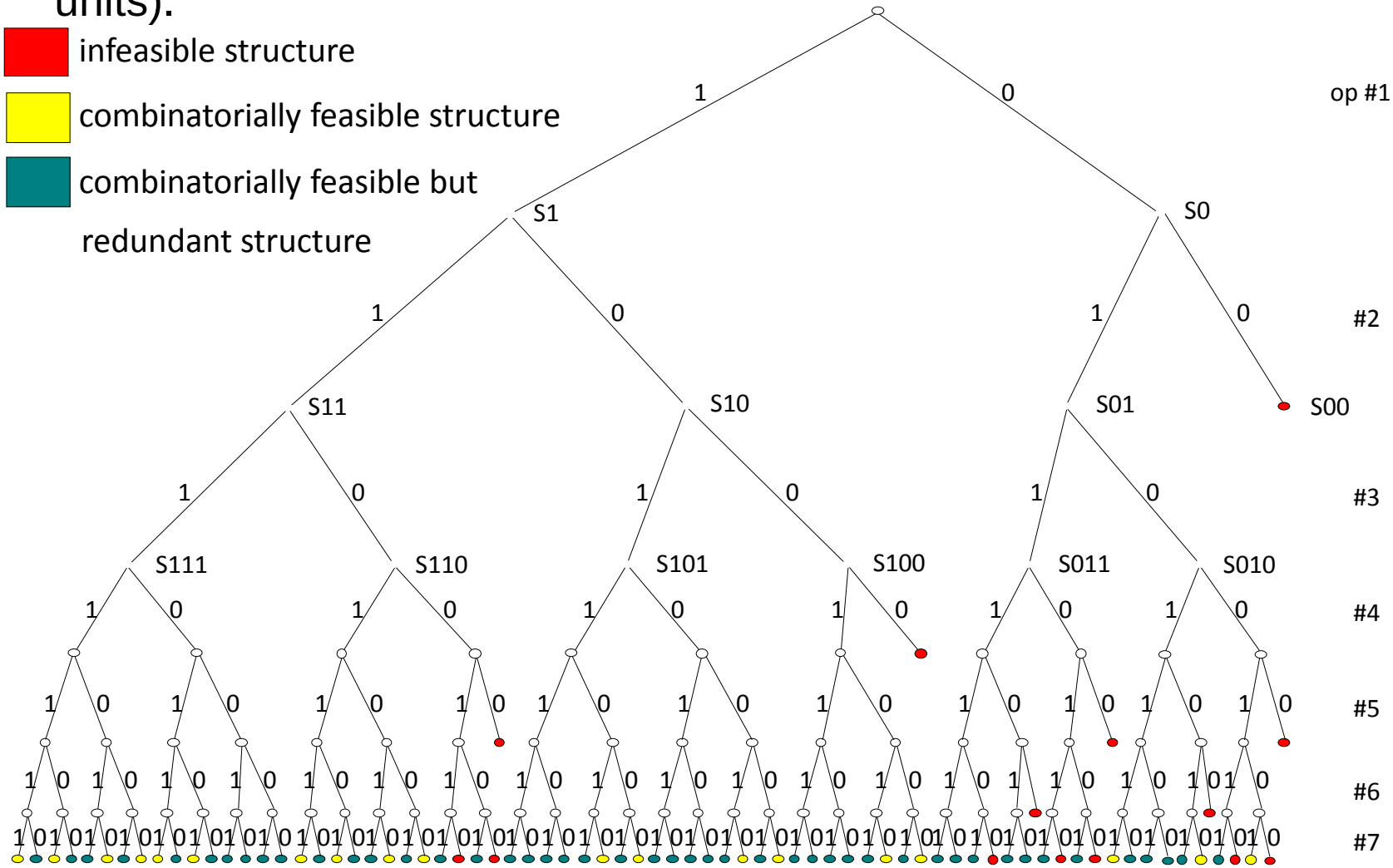
1	existence or inclusion of the corresponding operating unit
0	absence or exclusion of the corresponding operating unit

- Illustration of the inefficiency of the basic branch-and-bound algorithm for the worst case with a simple example: Example PNS 1 (7 operating units).

infeasible structure

combinatorially feasible structure

combinatorially feasible but redundant structure



Note: Each node of the tree represents one LP (or NLP) problem.

Note: In the worst case, 157 partial problems are examined to determine the optimal solution which is always among the 19 combinatorially feasible structures.

EXAMPLE PNS 2

(Industrial synthesis problem with 35 operating units)

Number of partial problems generated by the basic branch-and-bound algorithm:

130 million

Number of combinatorially feasible structures:

3465

Note: The large ratio shows high inefficiency.

GENERAL PROPERTIES OF THE BRANCH-AND-BOUND FRAMEWORK IN SOLVING PNS

The basic branch-and-bound algorithm is inefficient in solving a process synthesis problem because:

- it leads to a large number of partial problems,
- each partial problem has an unnecessarily, large number of free variables.

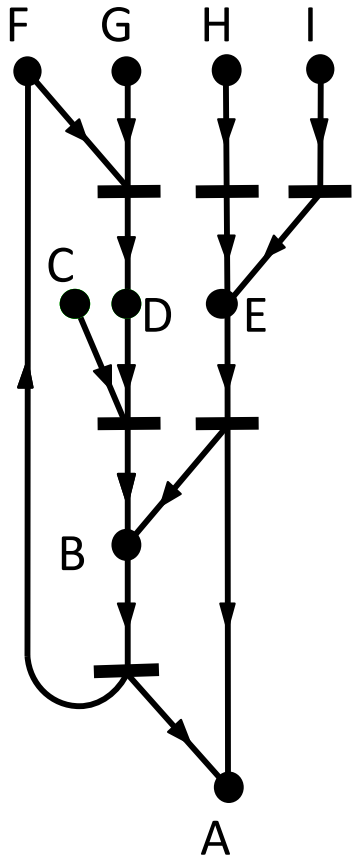
ACCELERATED BRANCH AND BOUND ALGORITHM

The accelerated branch-and-bound algorithm

- reduce the size of an individual subproblem through exclusion of those operating units that should not be included in any feasible solution of the subproblem
- speeds up the generation of the optimal solution by minimizing the number of subproblems to be solved

ILLUSTRATIVE EXAMPLE FOR BRANCHING BY ABB

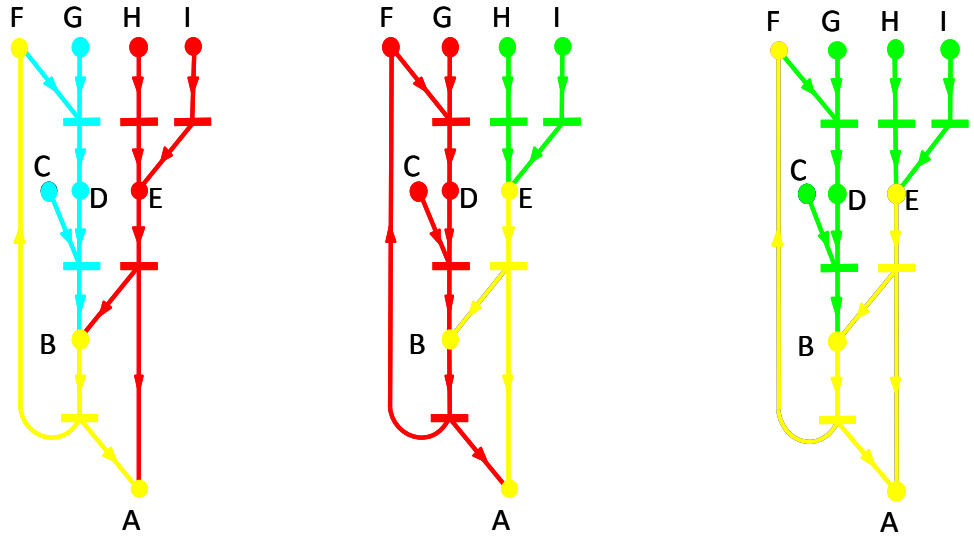
- Maximal structure



Product: A

Raw materials: C, F, G, H, I

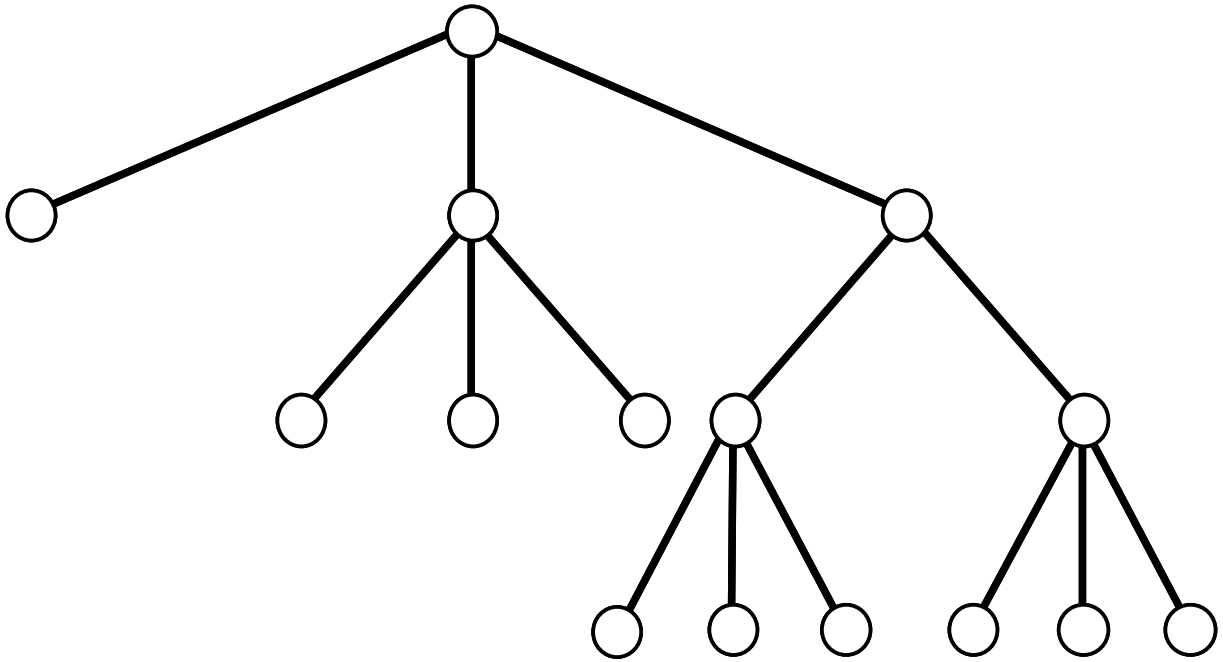
PARTIAL PROBLEMS GENERATED ON THE BASIS OF THE PRODUCTION OF A



Relation between the operating units and a partial problem

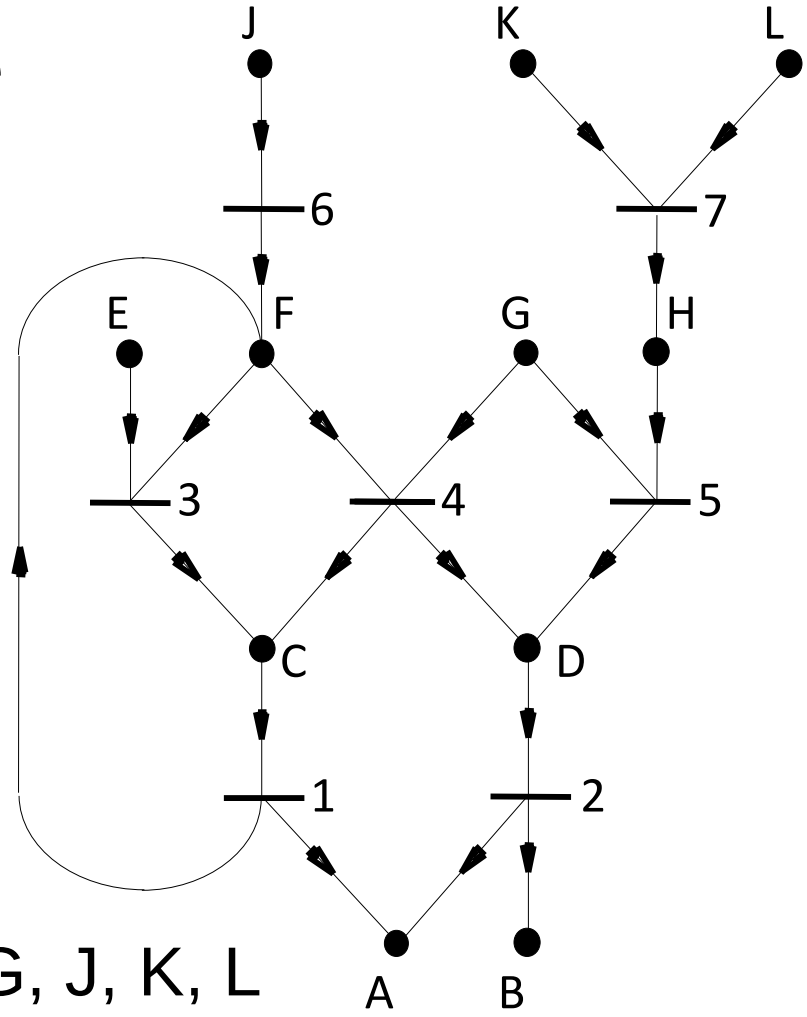
- included in each structure (based on decisions)
- included in each structure (based on maximal neutral extension)
- excluded from each structure
- included in at least one structure

ENUMERATION TREE FOR THE ACCELERATED BRANCH-AND-BOUND (WORST CASE)



ILLUSTRATIVE EXAMPLE FOR BRANCHING BY ABB

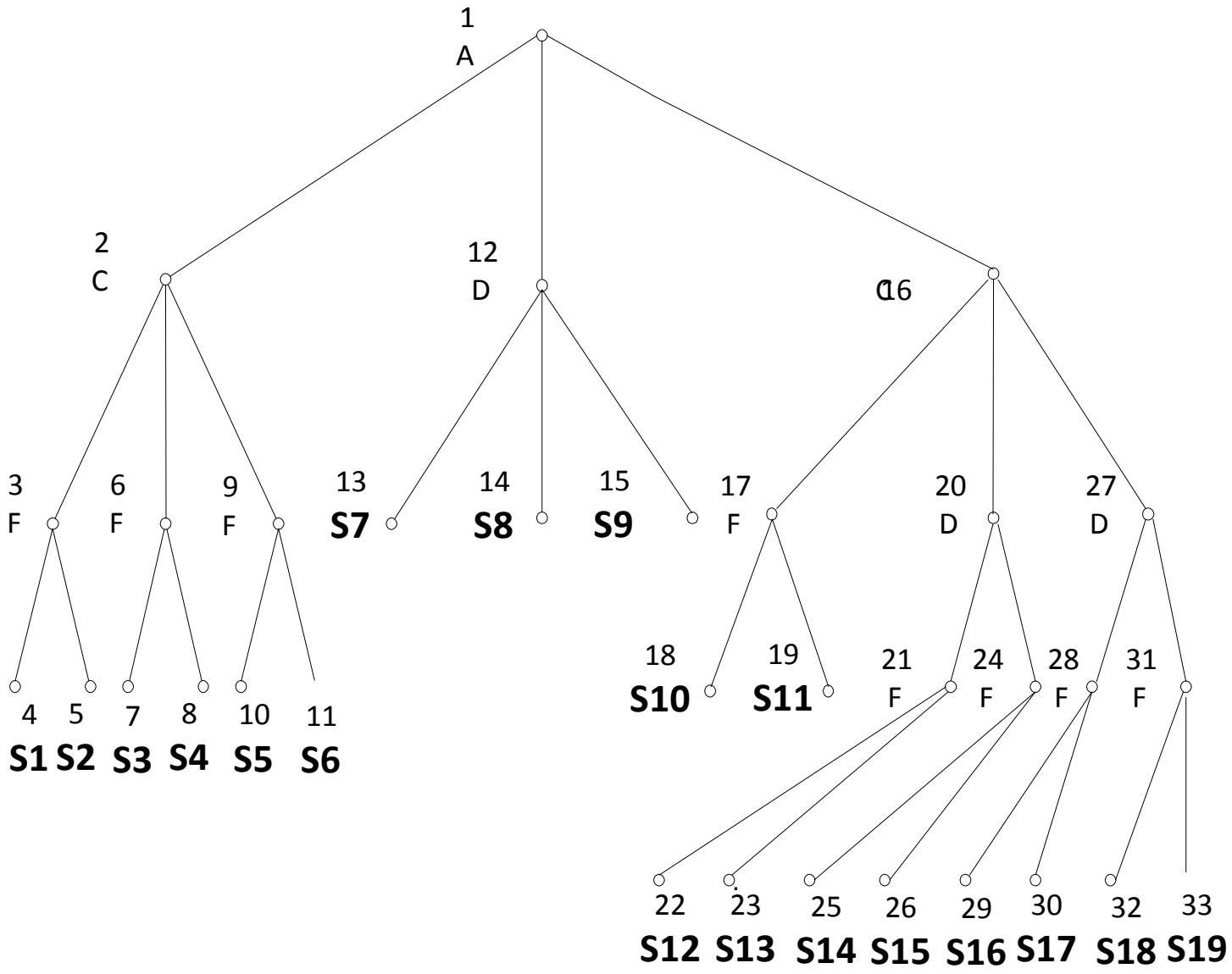
- Maximal structure



Product: A

Raw materials: E, G, J, K, L

ENUMERATION TREE (WORST CASE)



- Relation between the operating units and a partial problem.

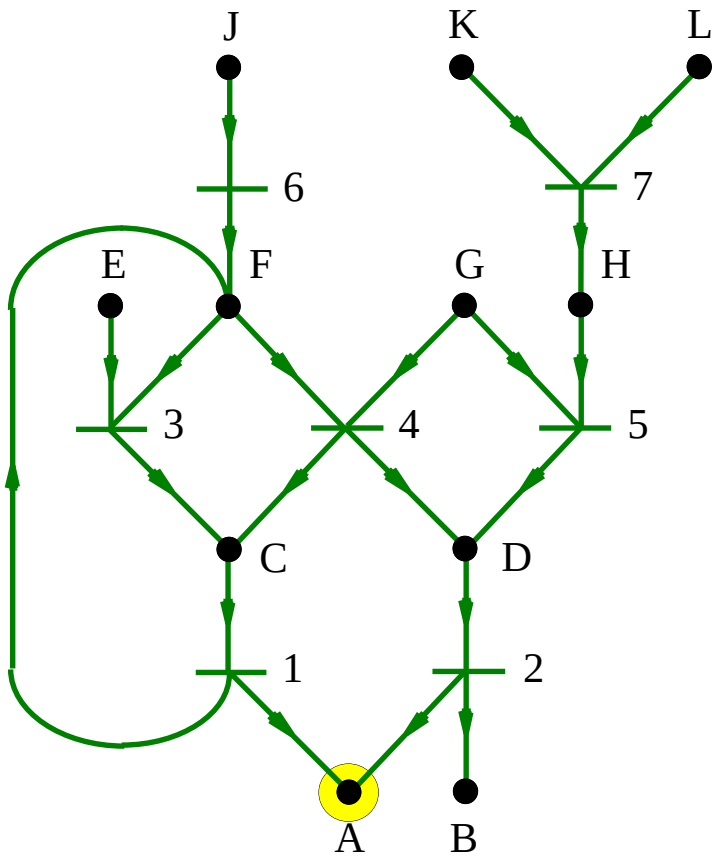
included in each structure
(based on decisions)

included in each structure
(based on maximal neutral extension)

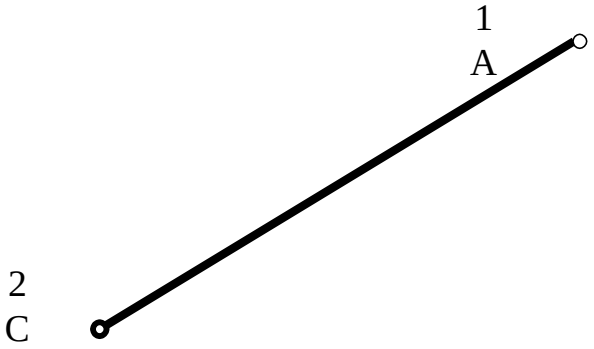
excluded from each structure

included in at least one structure

Partial problem #1

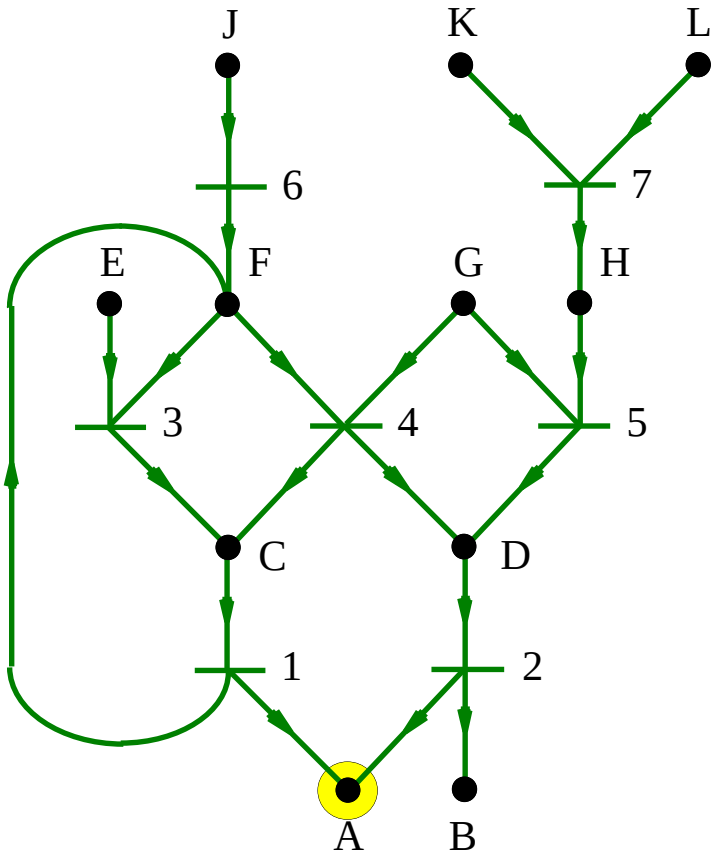


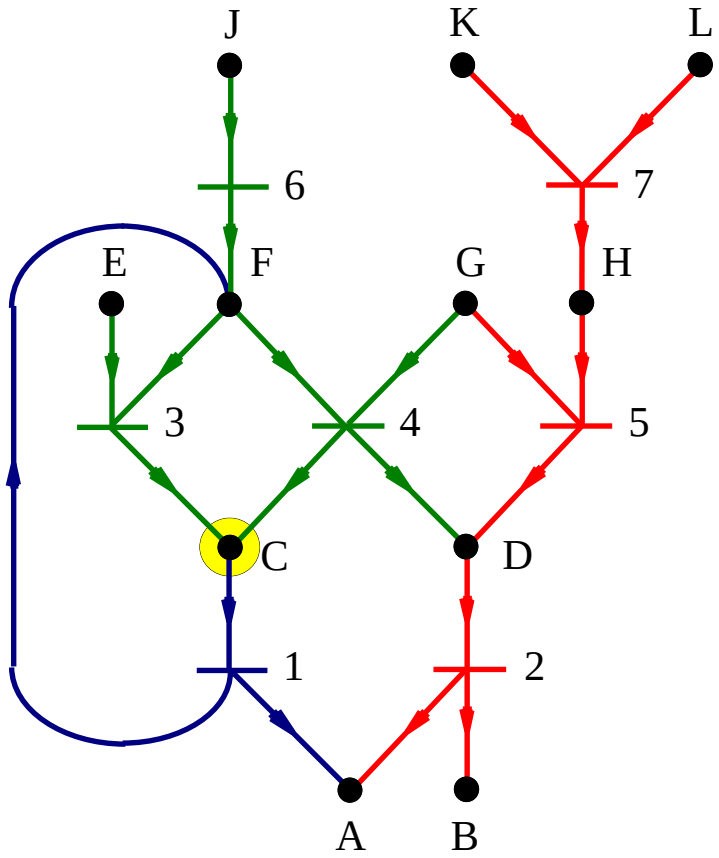
Enumeration (search) tree (worst case)



Si: solution
Number: partial problem
Capital letter: decision point on the maximal structure

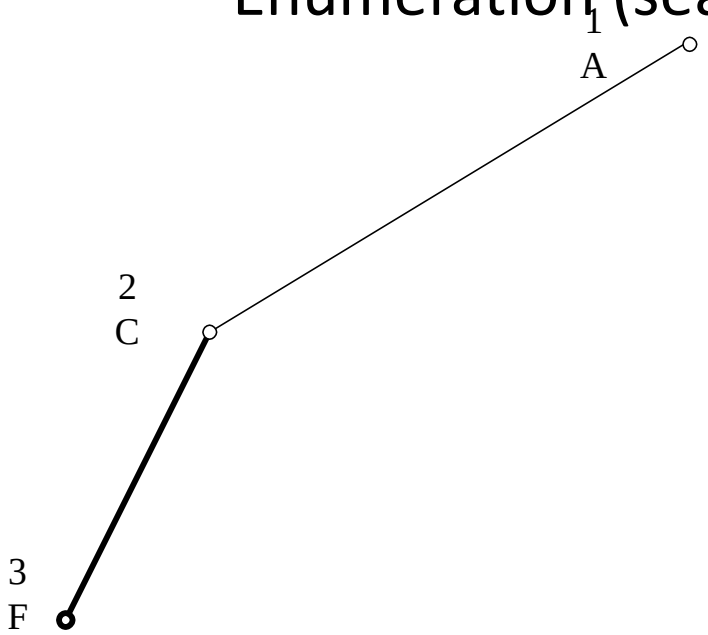
Partial problem #1





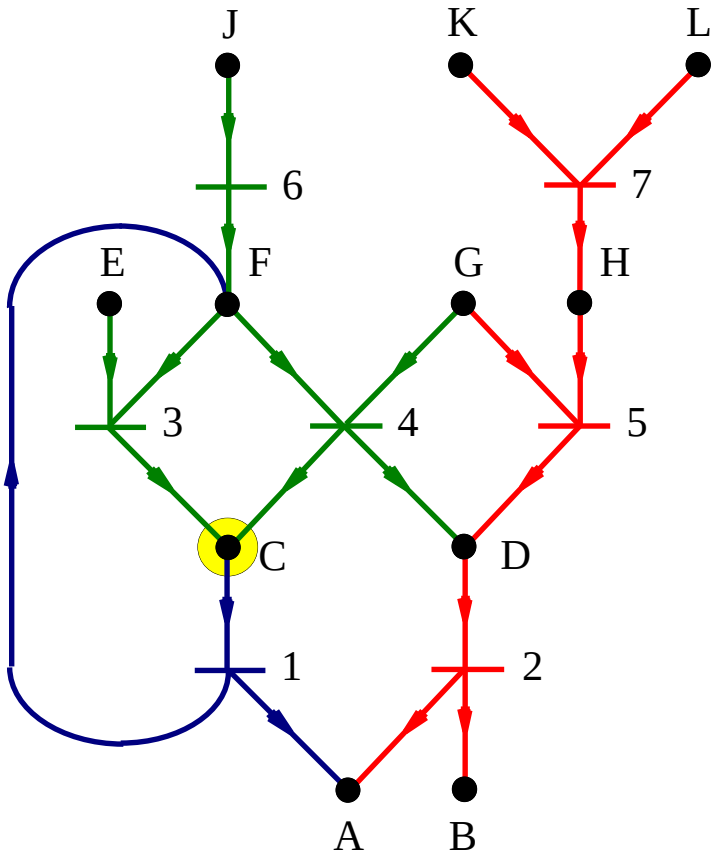
Partial problem #2

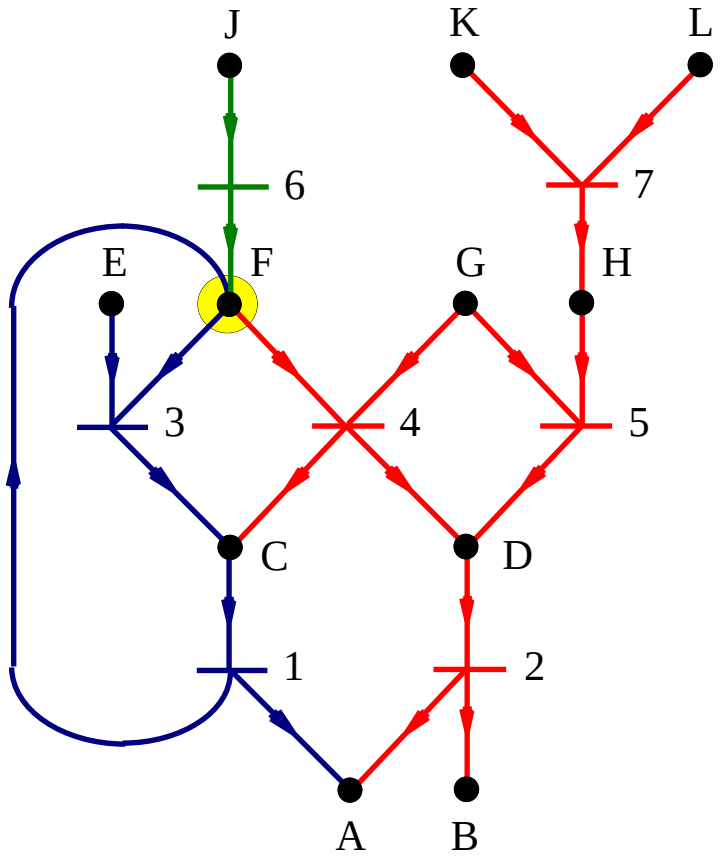
Enumeration (search) tree (worst case)



Si: solution
Number: partial problem
Capital letter: decision point on the maximal structure

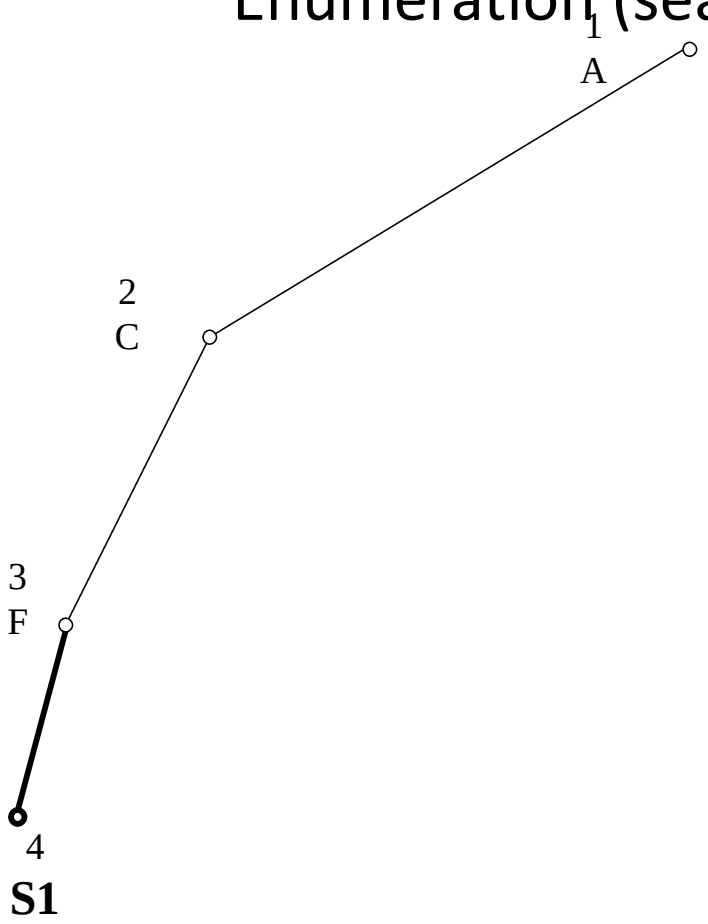
Partial problem #2





Partial problem #3

Enumeration (search) tree (worst case)

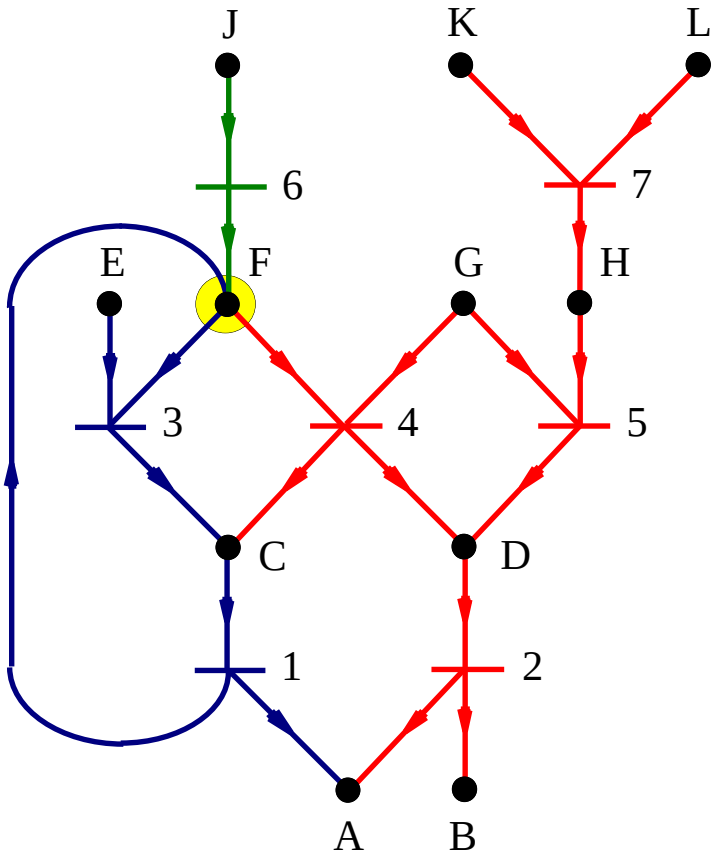


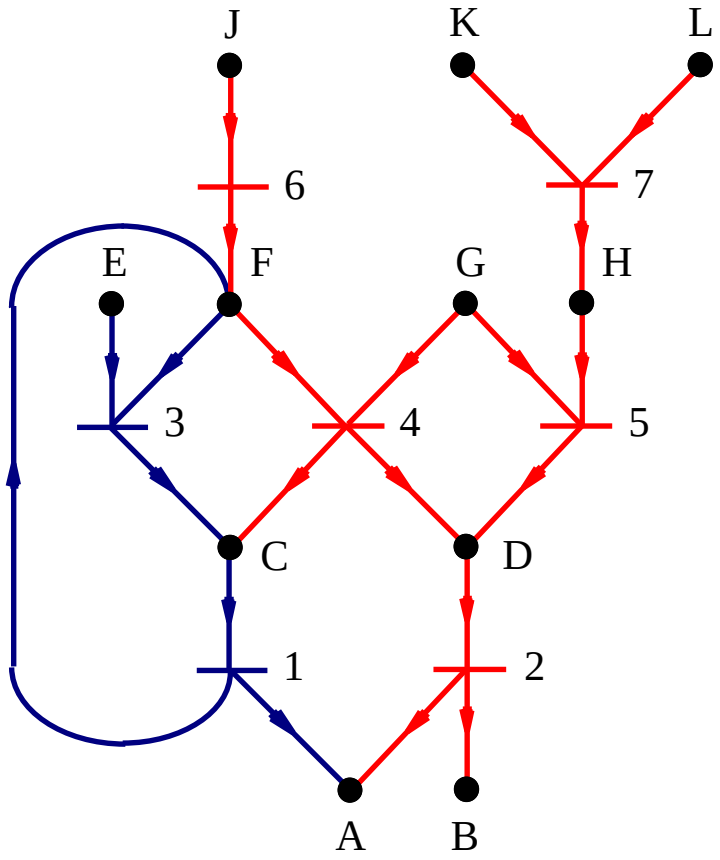
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

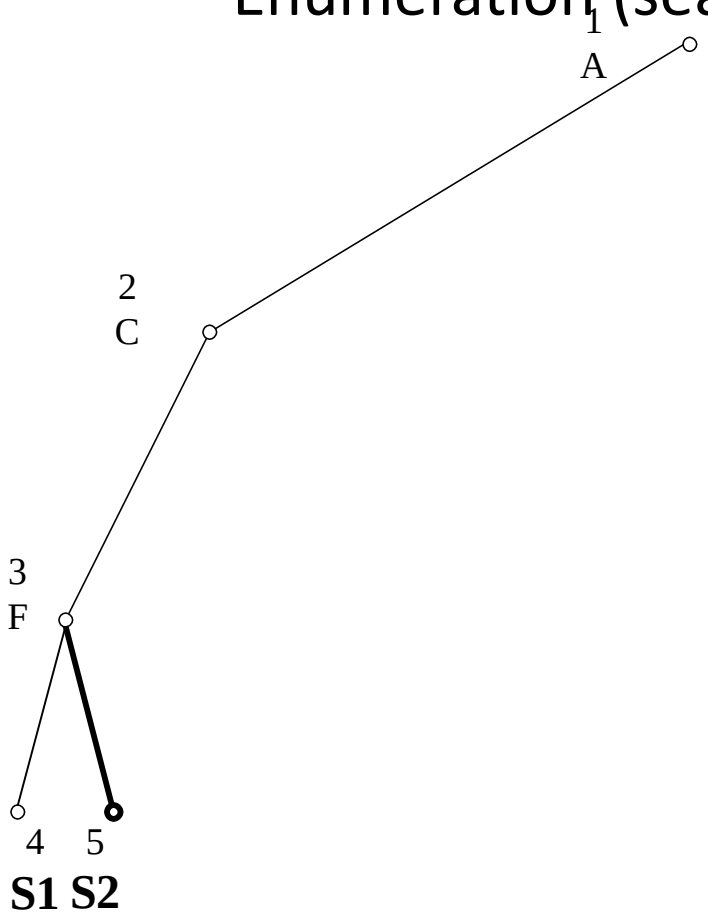
Partial problem #3





Partial problem #4 (S1)

Enumeration (search) tree (worst case)

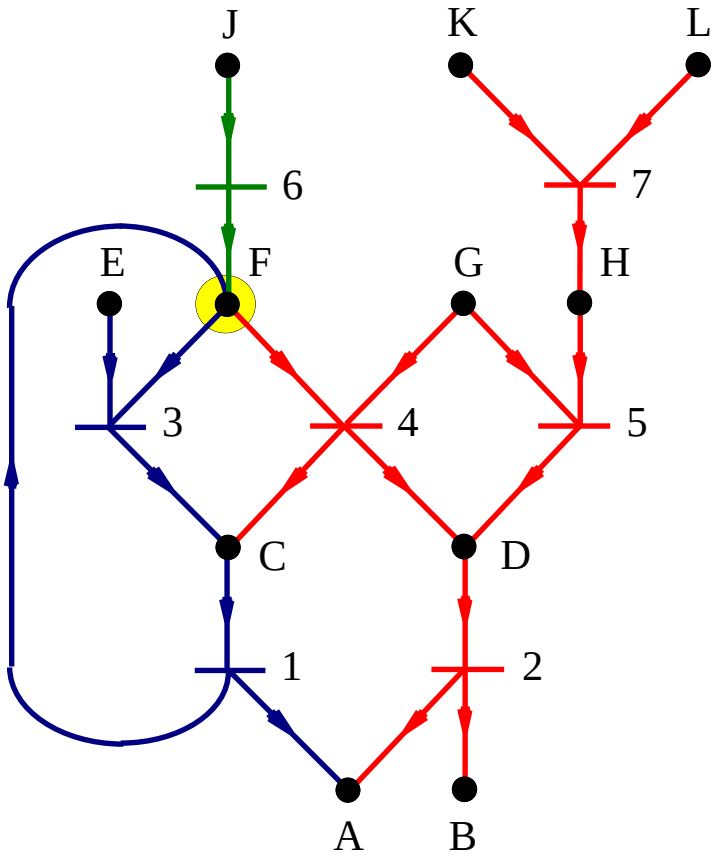


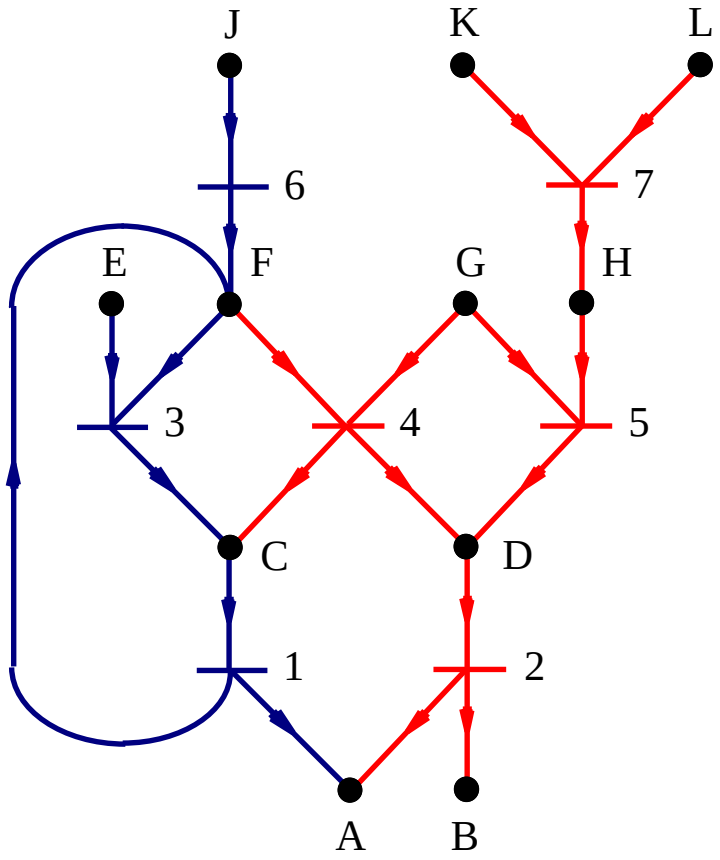
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

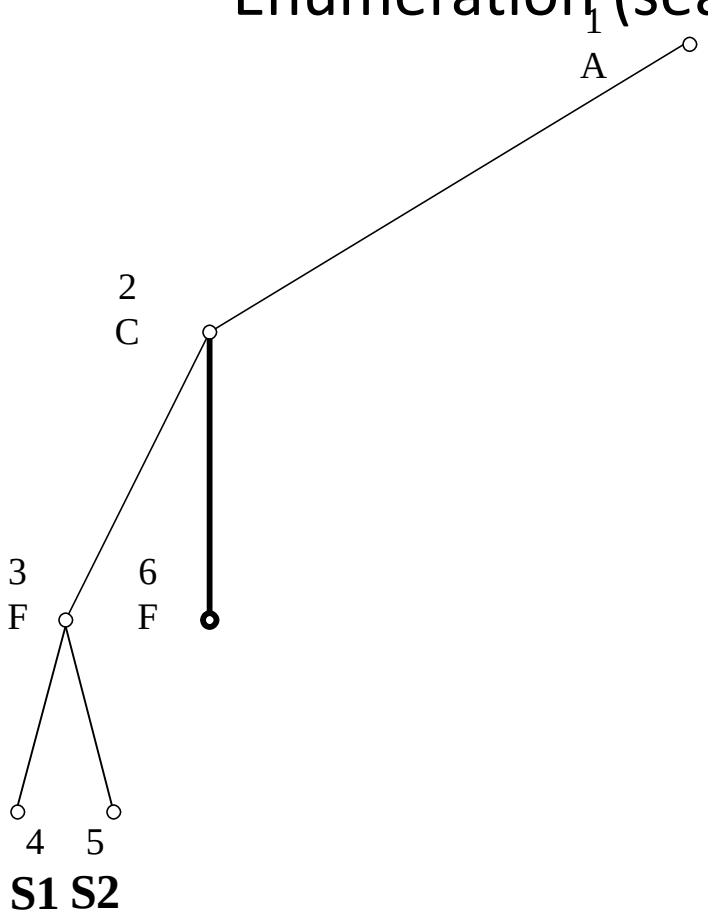
Partial problem #3





Partial problem #5 (S2)

Enumeration (search) tree (worst case)

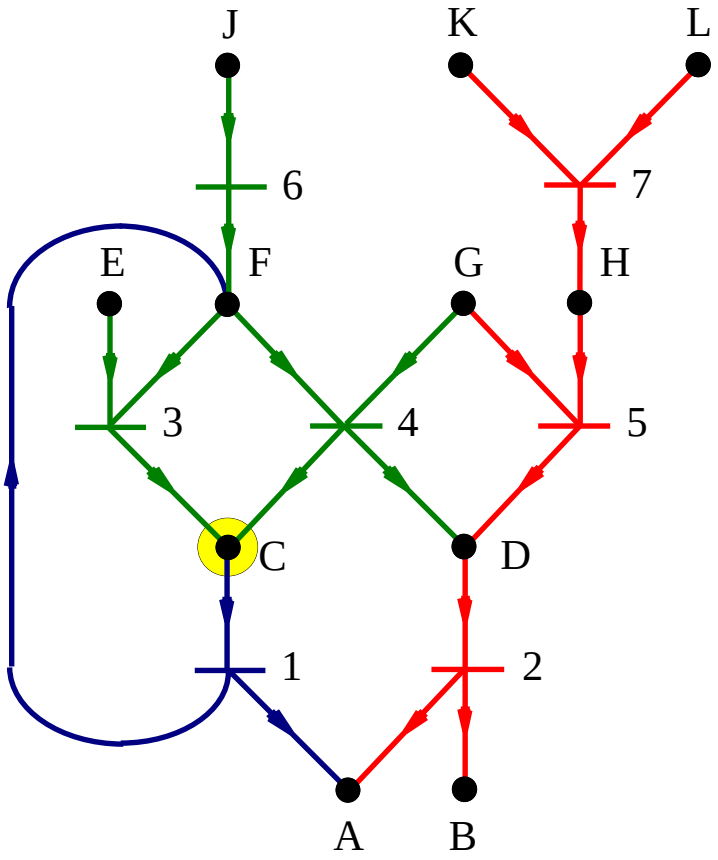


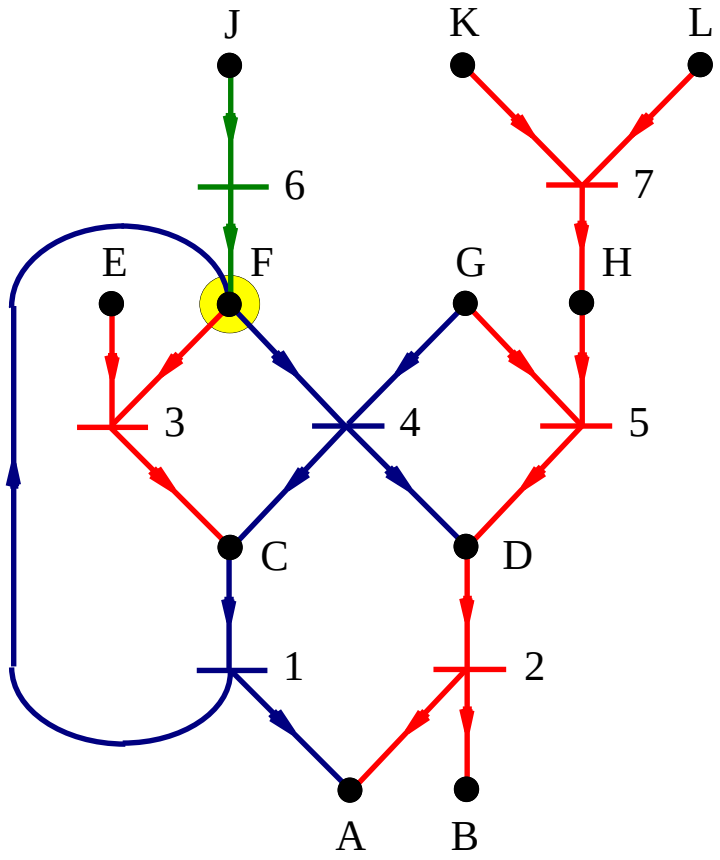
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

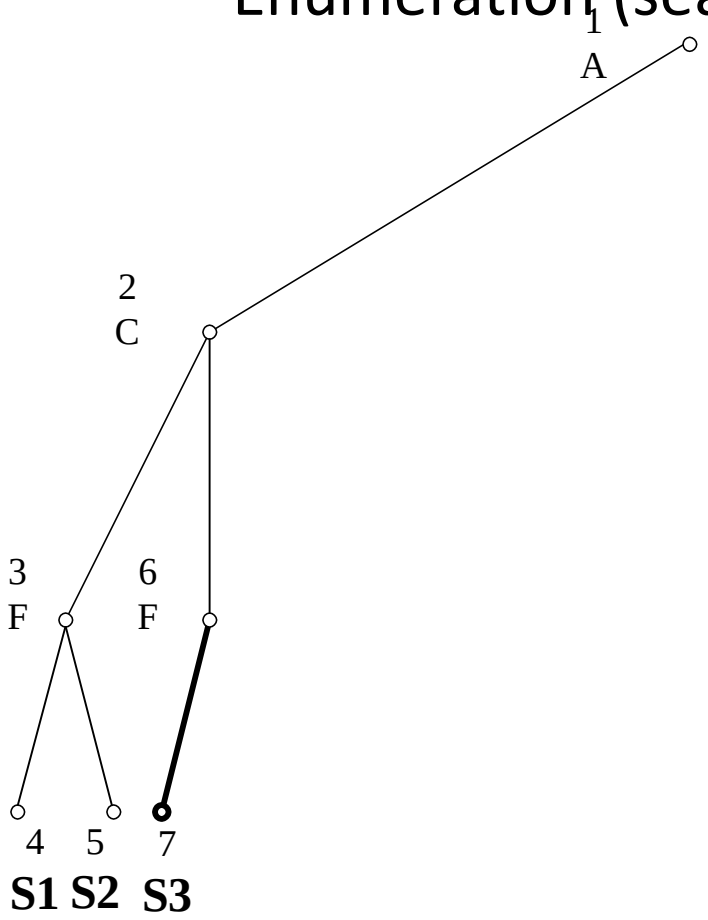
Partial problem #2





Partial problem #6

Enumeration (search) tree (worst case)

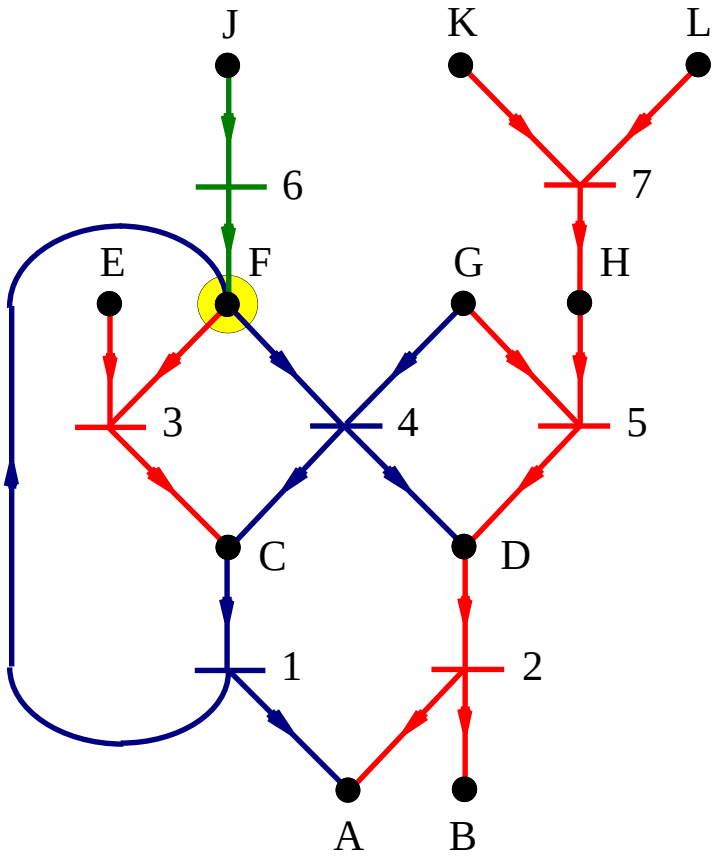


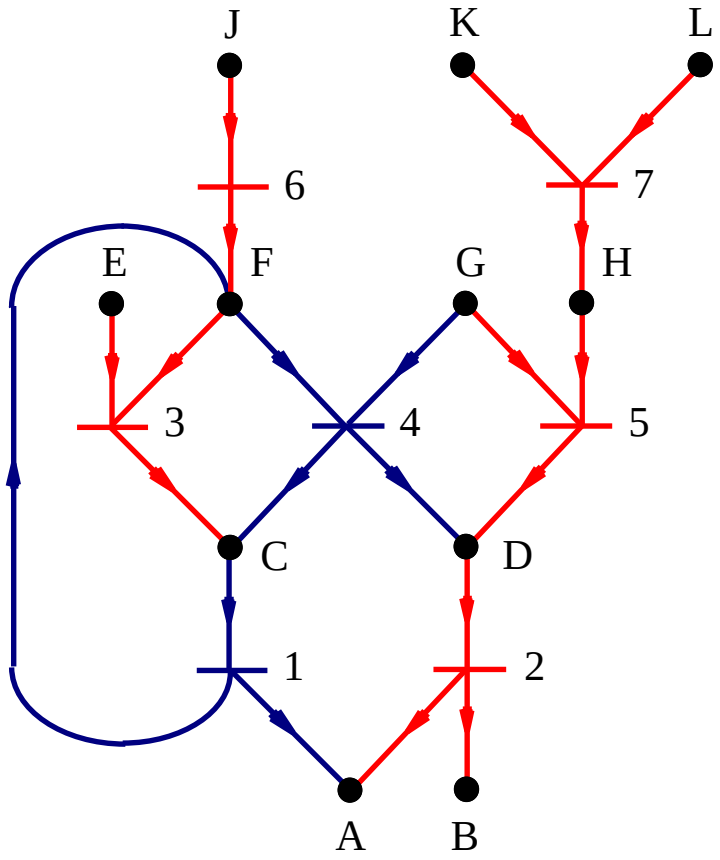
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

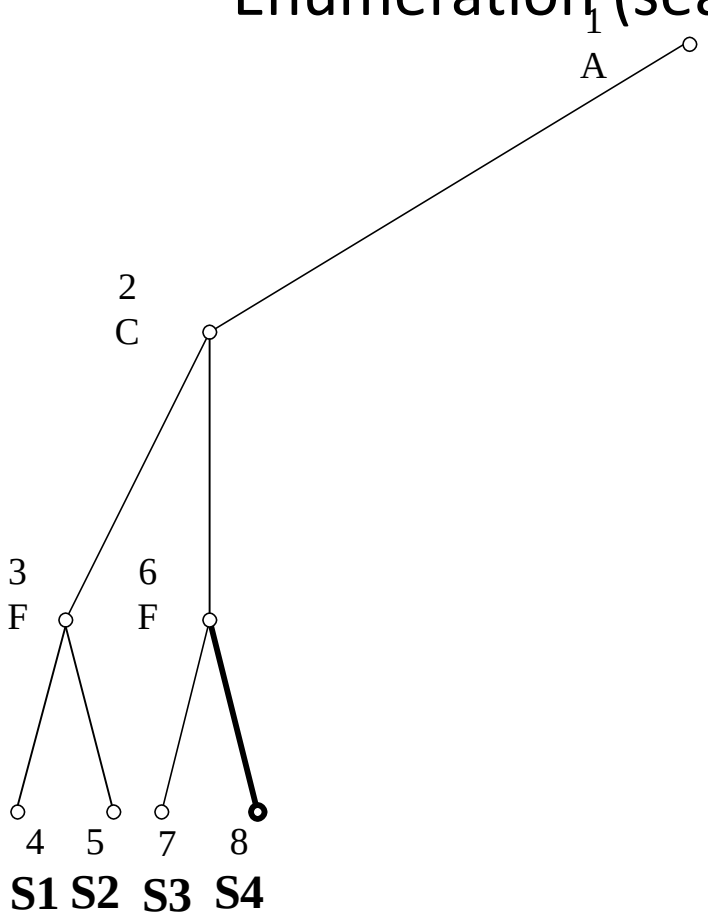
Partial problem #6





Partial problem #7 (S3)

Enumeration (search) tree (worst case)

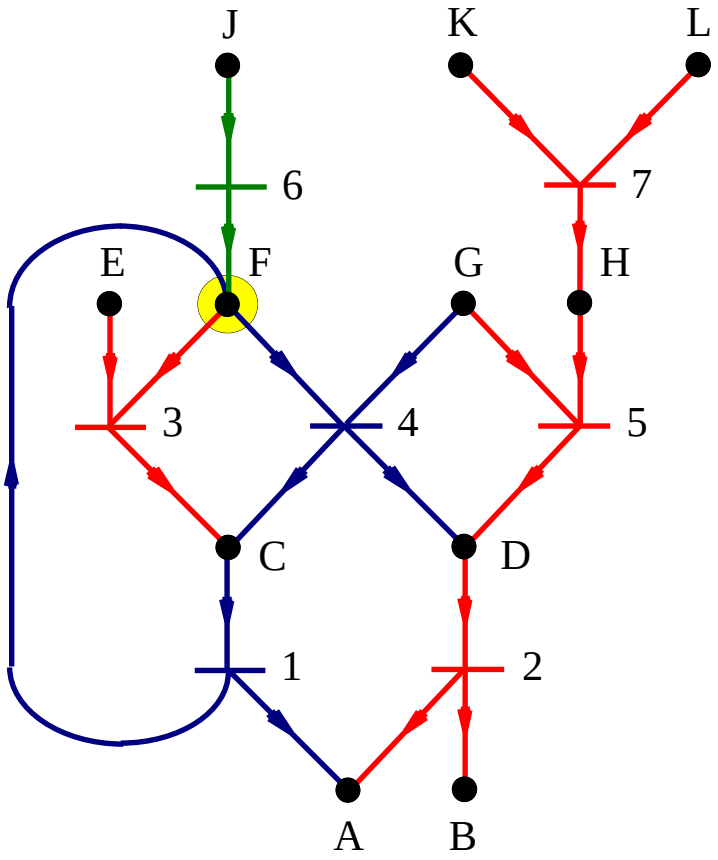


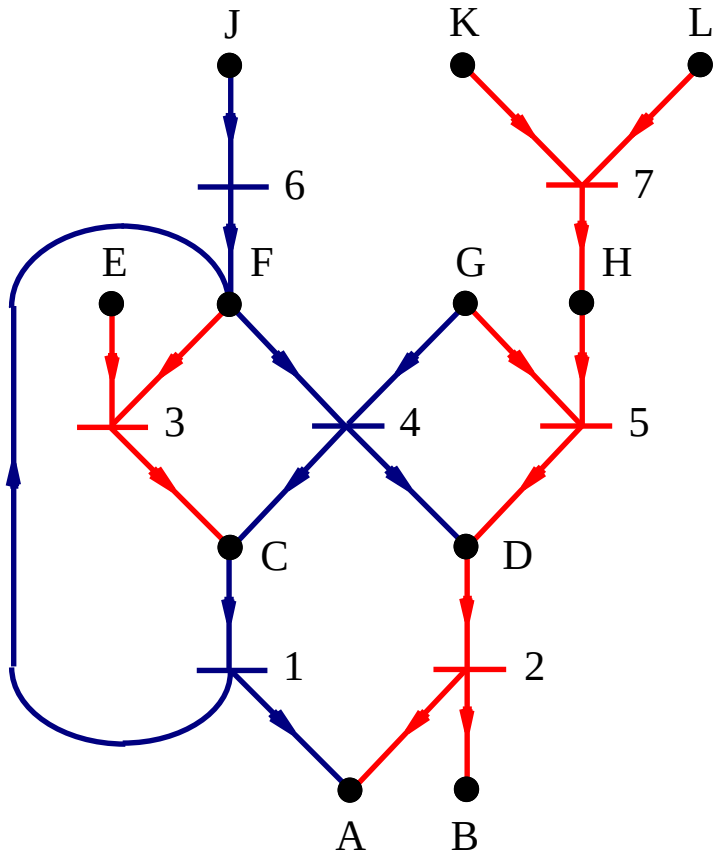
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

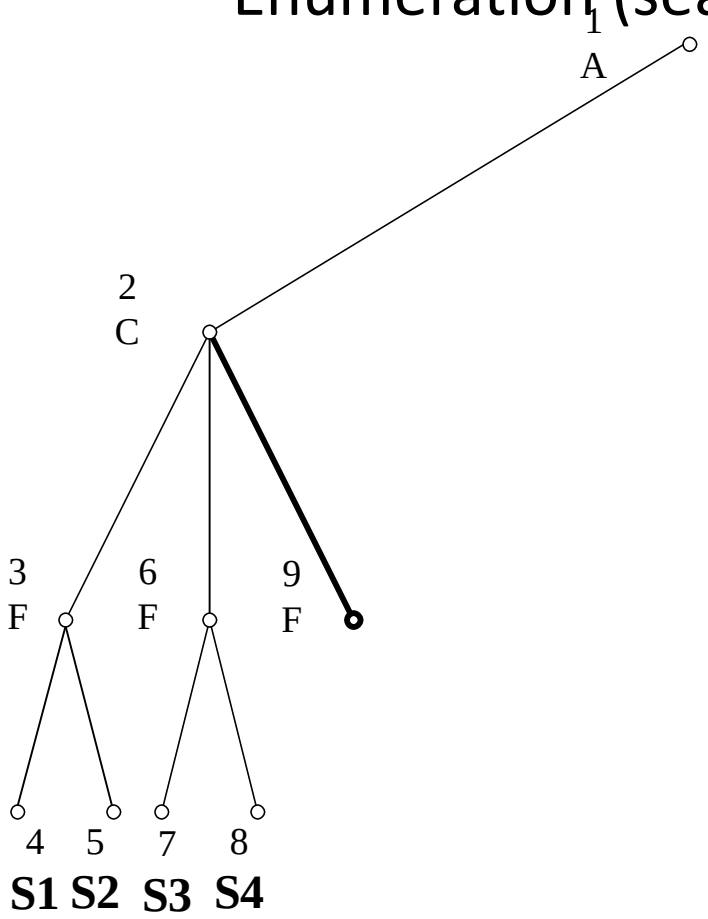
Partial problem #6





Partial problem #8 (S4)

Enumeration (search) tree (worst case)

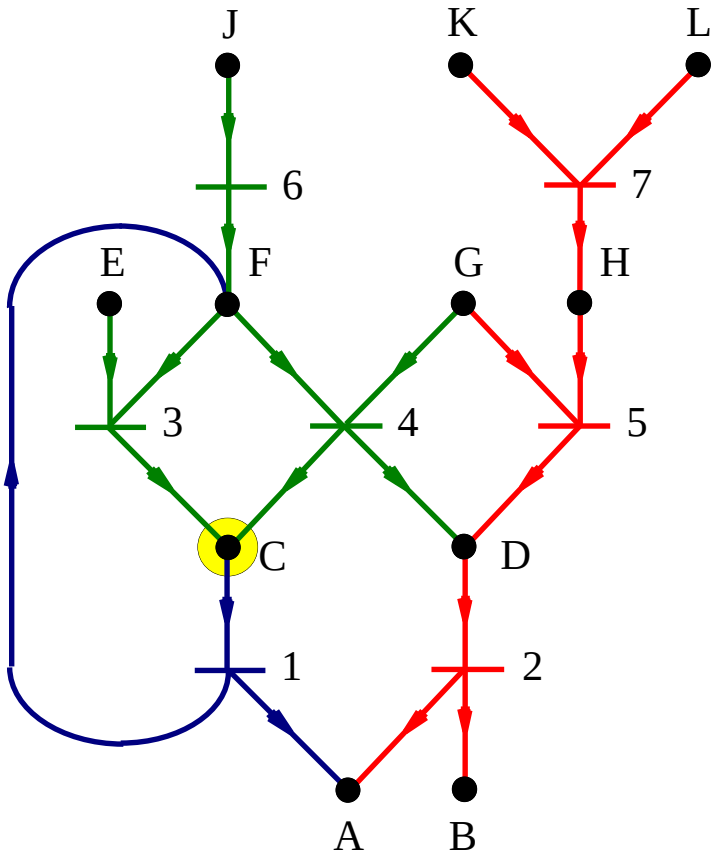


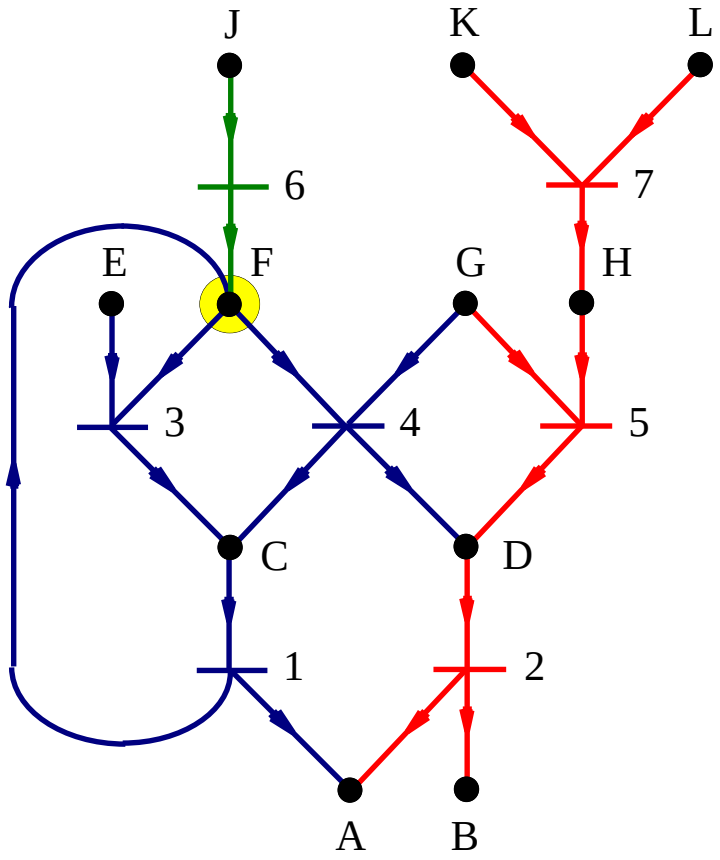
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

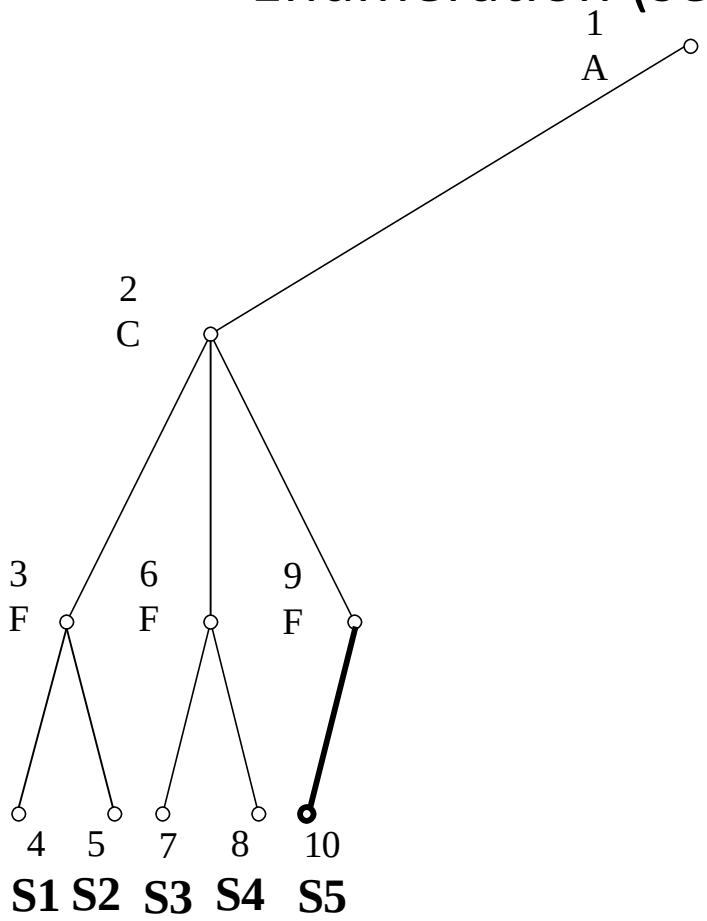
Partial problem #2





Partial problem #9

Enumeration (search) tree (worst case)

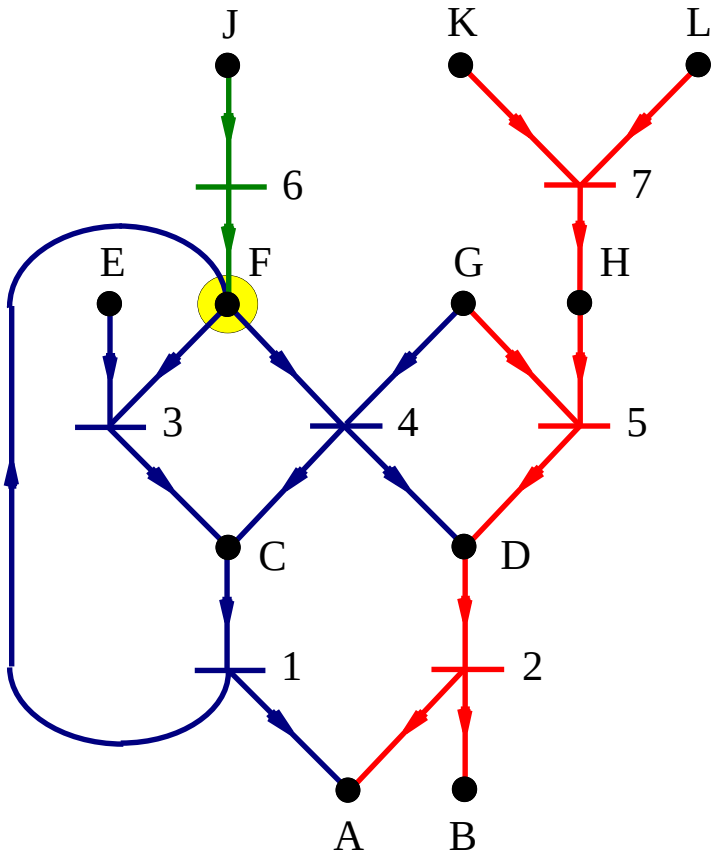


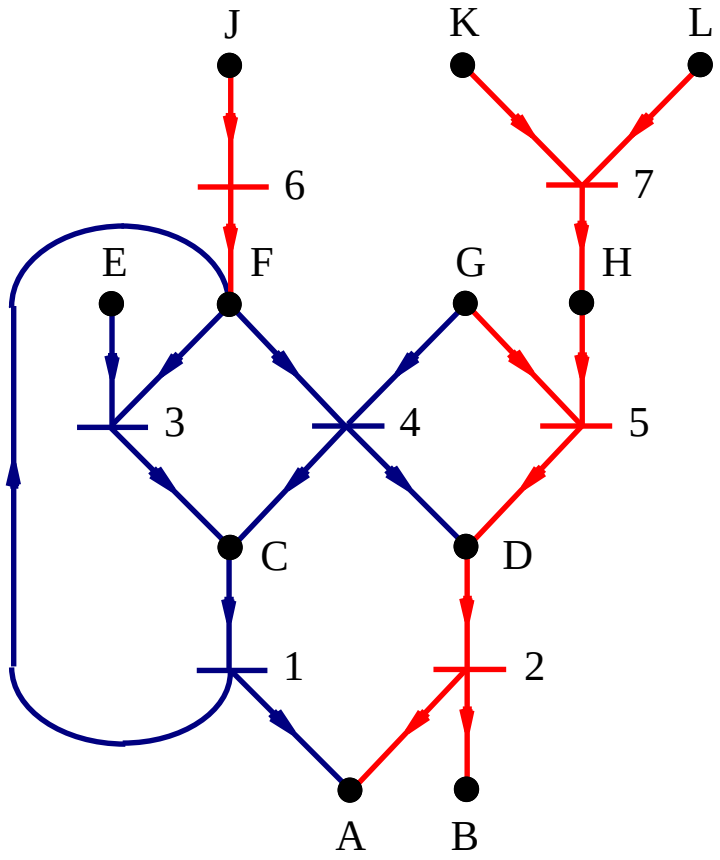
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

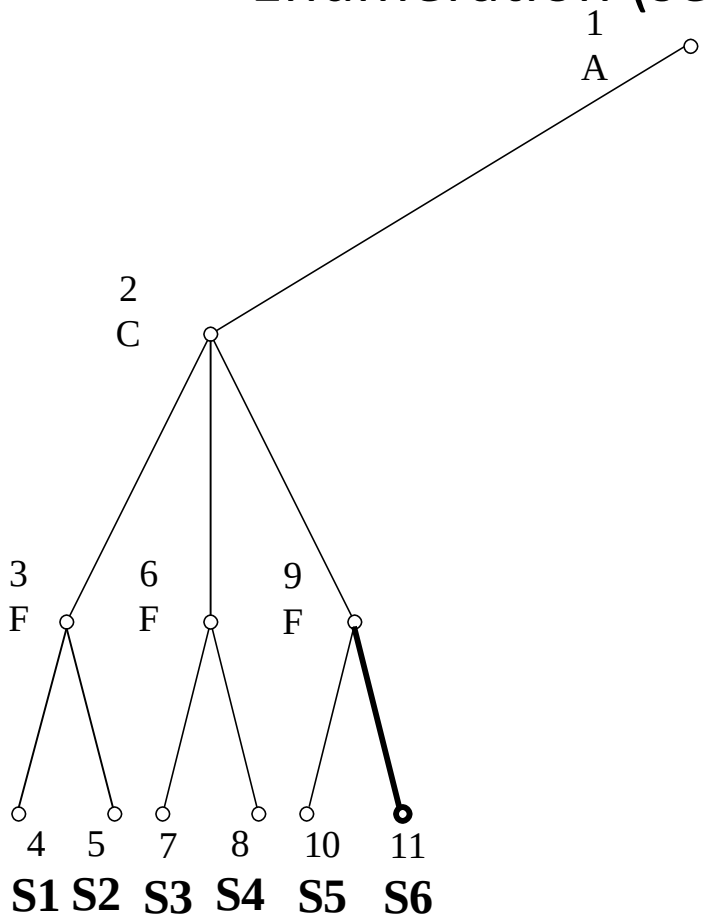
Partial problem #9





Partial problem #10 (S5)

Enumeration (search) tree (worst case)

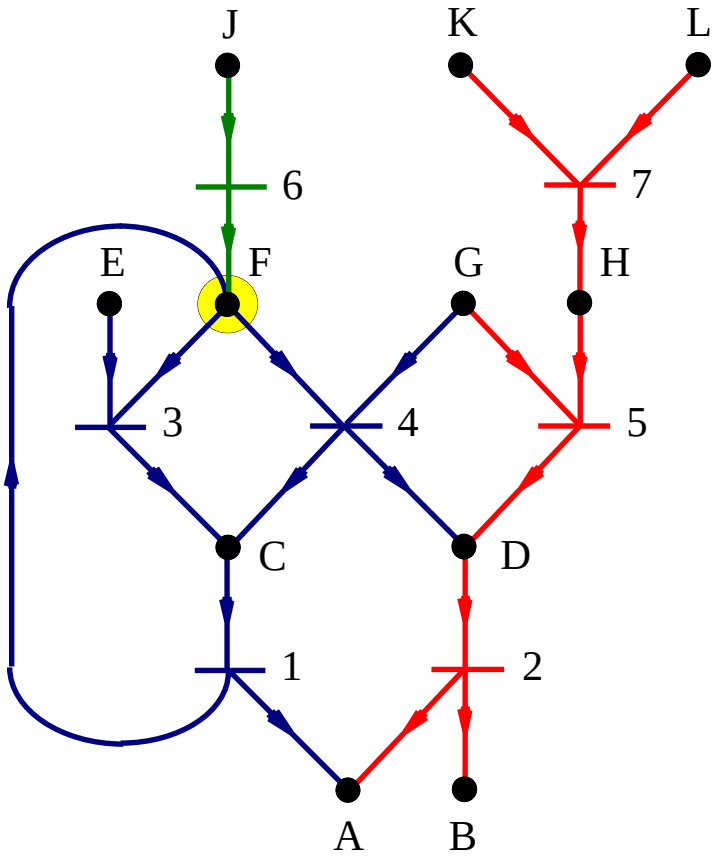


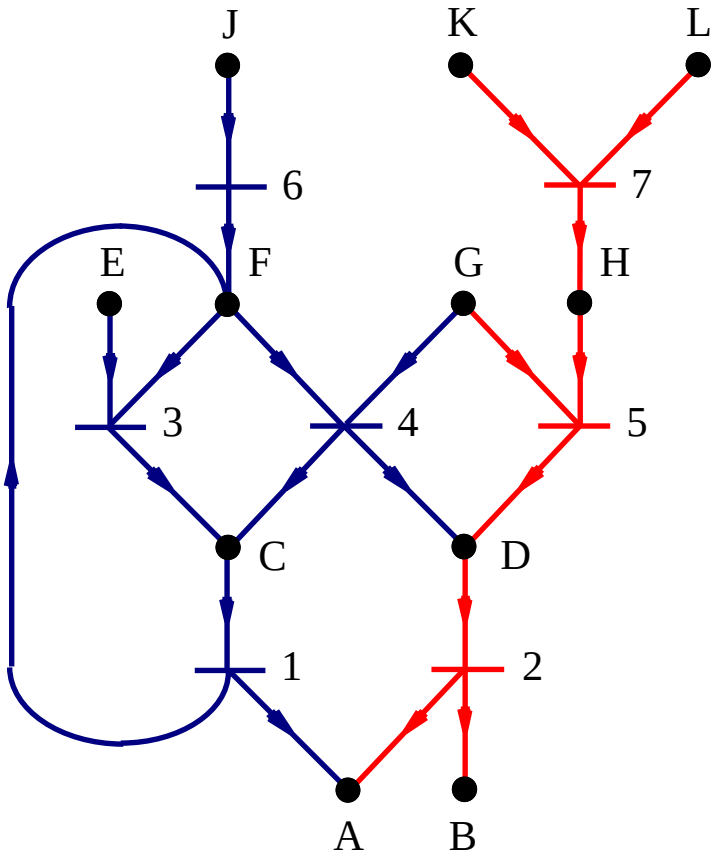
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

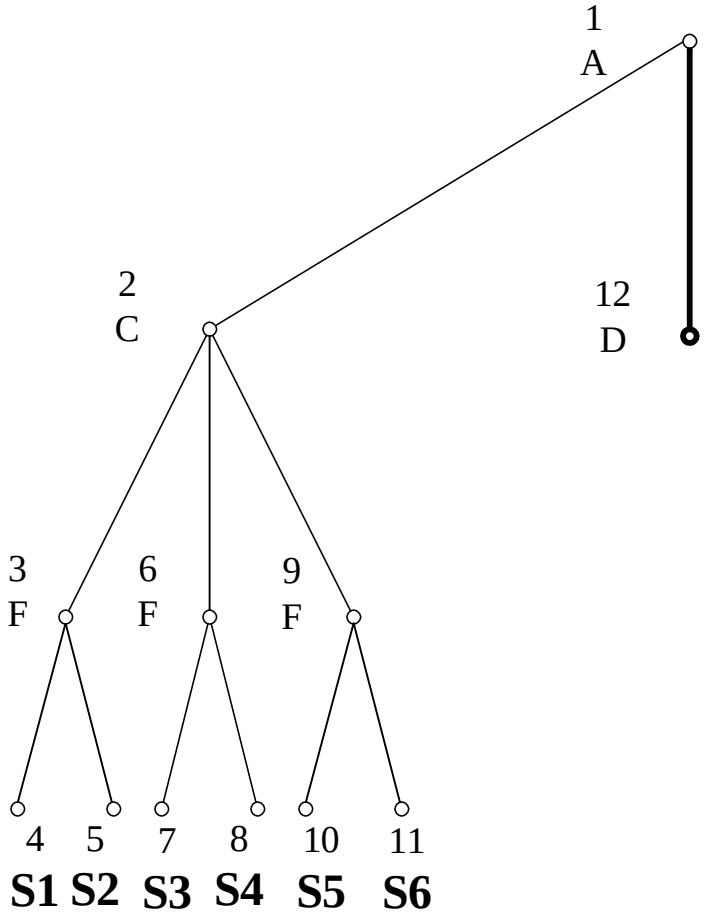
Partial problem #9





Partial problem #11 (S6)

Enumeration (search) tree (worst case)

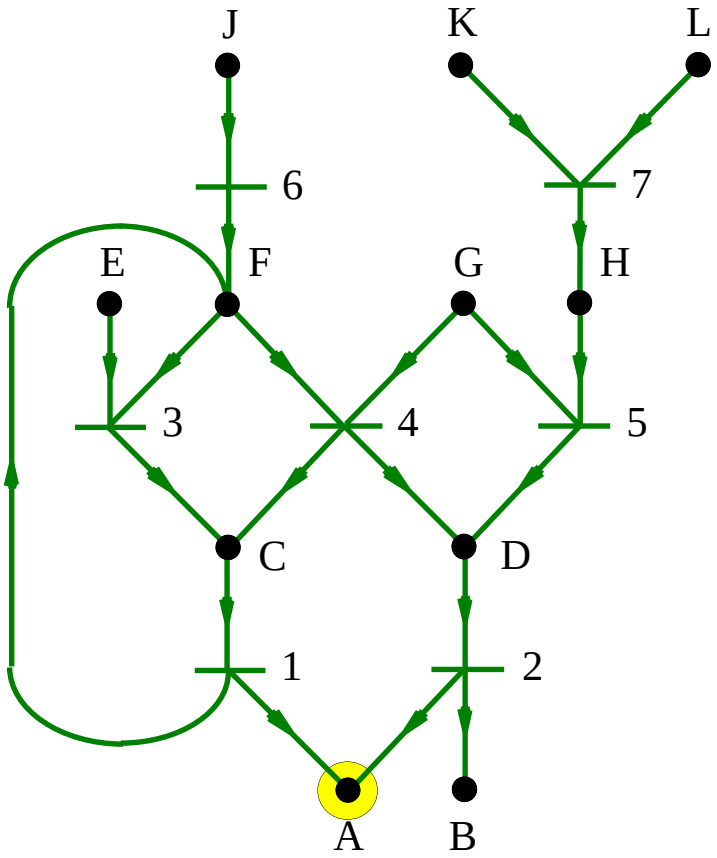


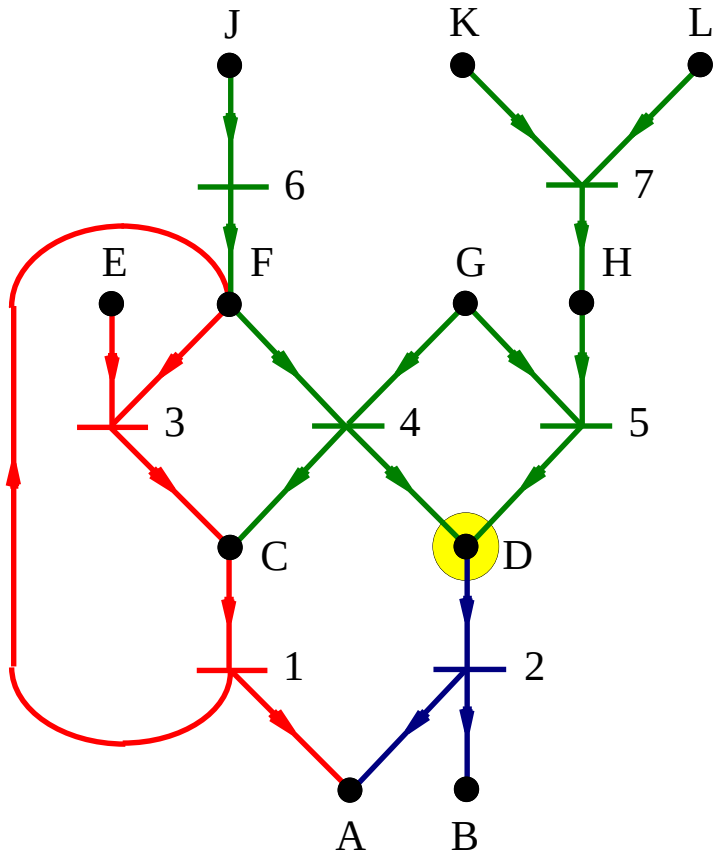
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

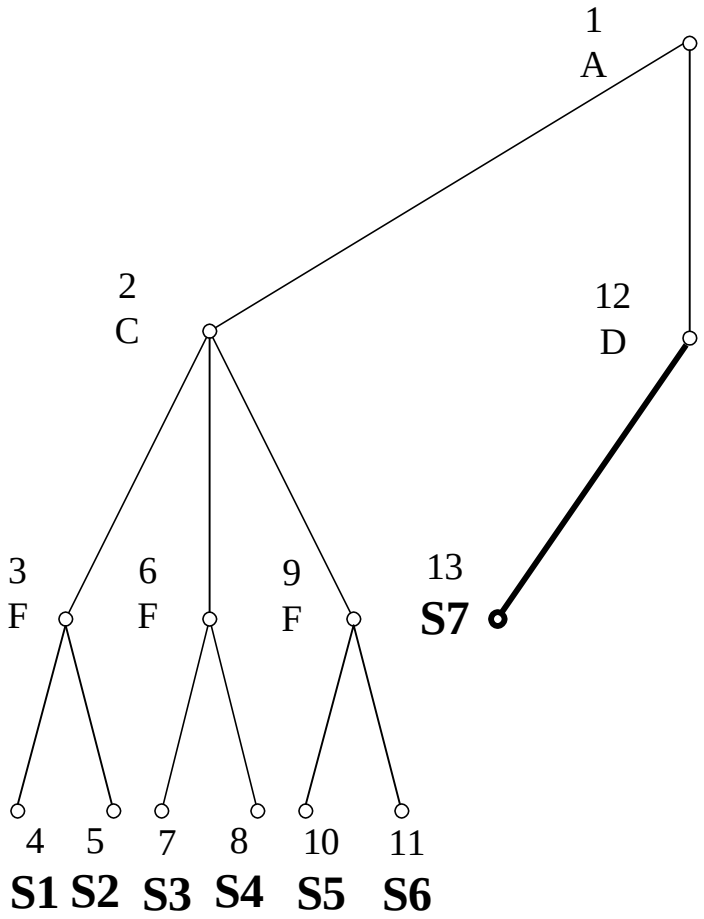
Partial problem #1





Partial problem #12

Enumeration (search) tree (worst case)

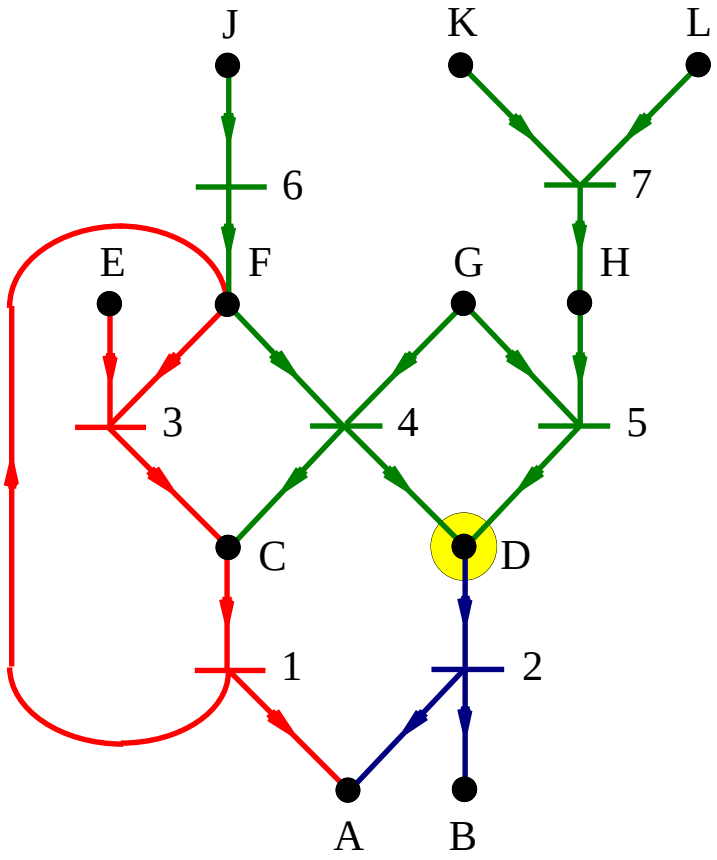


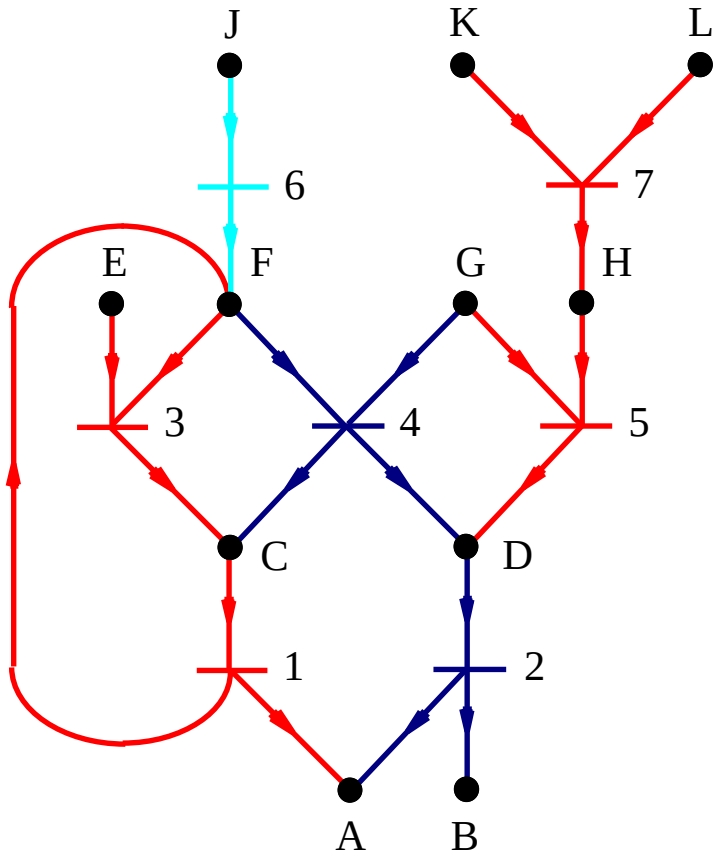
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

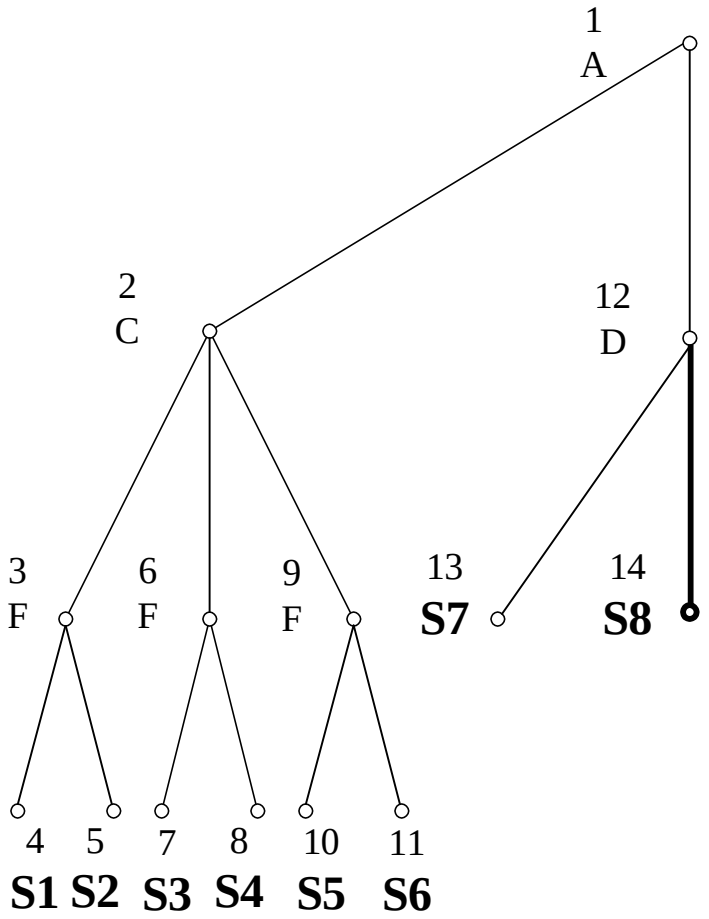
Partial problem #12





Partial problem #13 (S7)

Enumeration (search) tree (worst case)

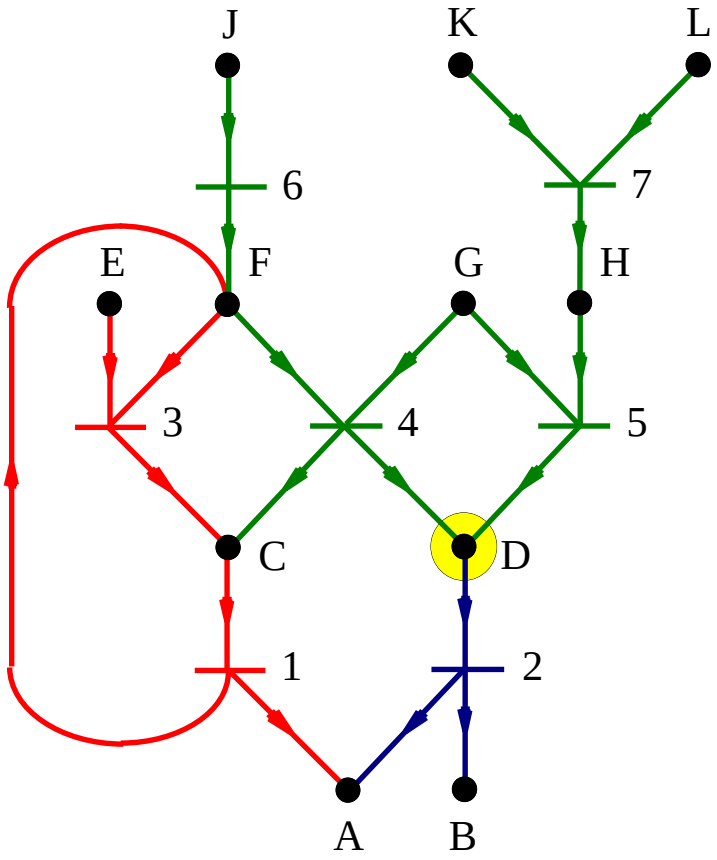


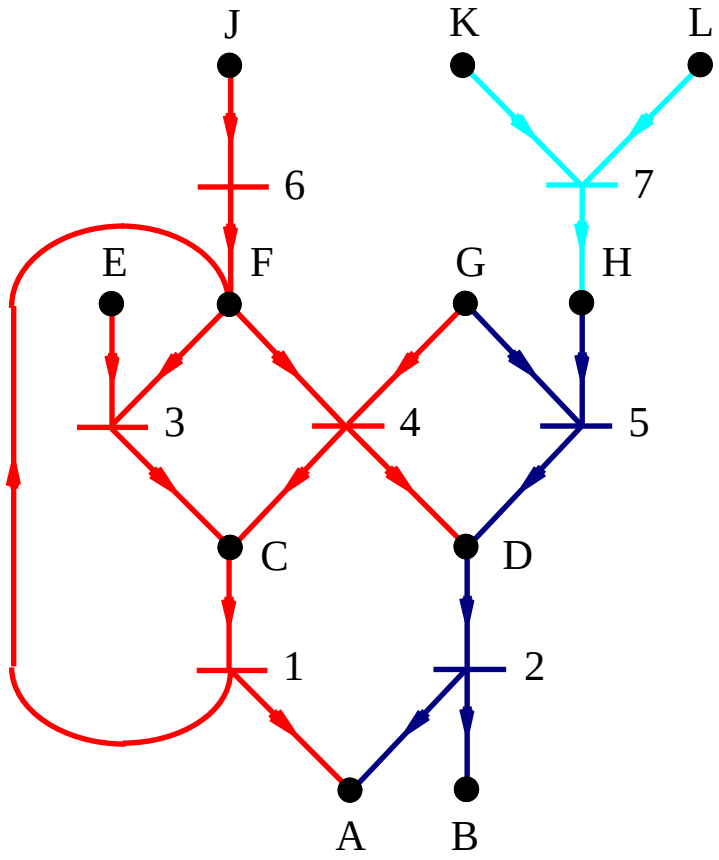
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

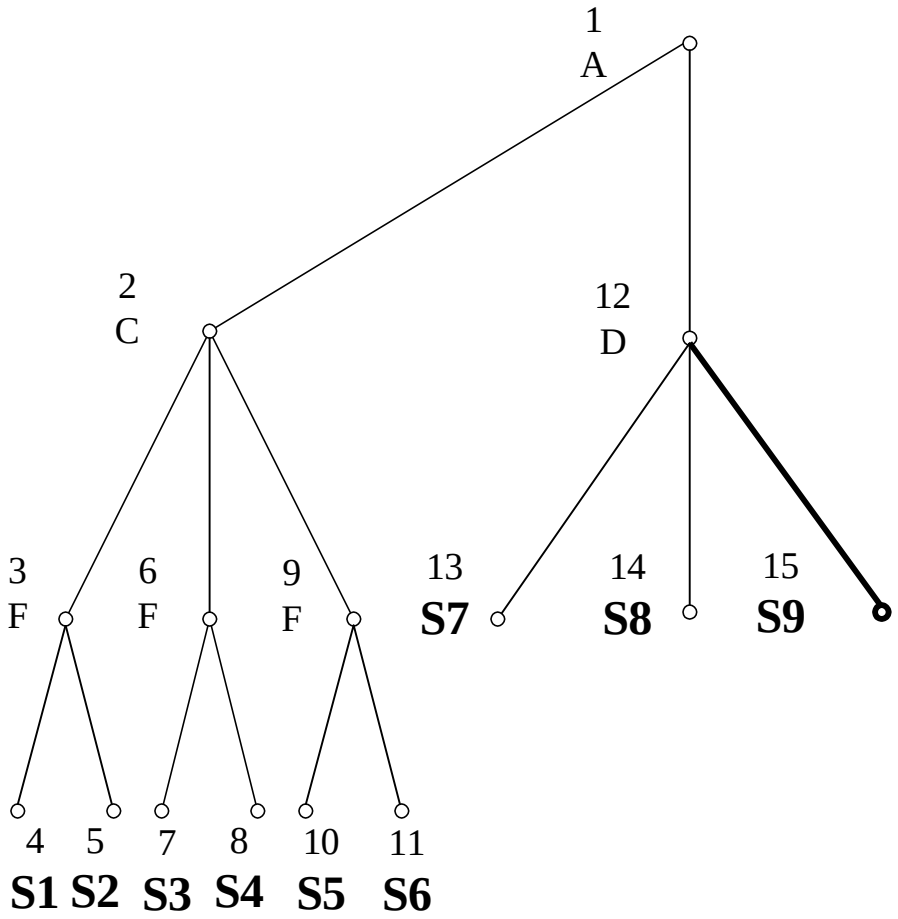
Partial problem #12





Partial problem #14 (S8)

Enumeration (search) tree (worst case)

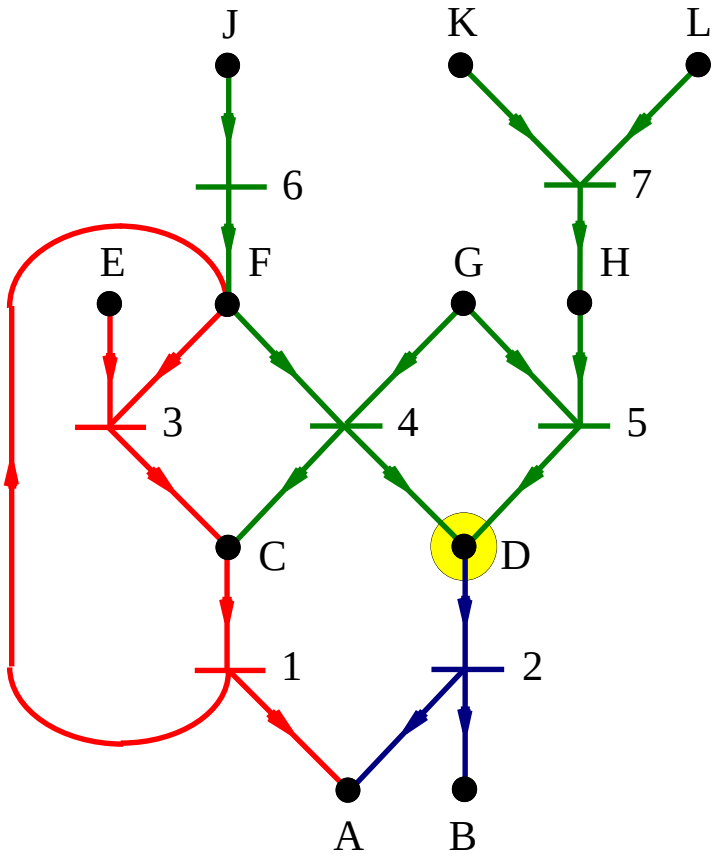


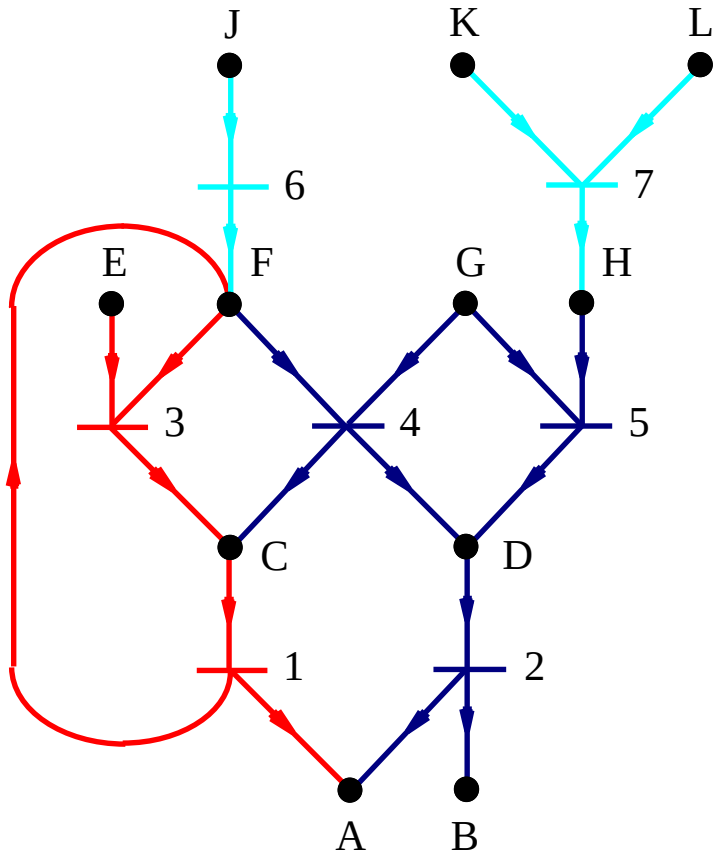
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

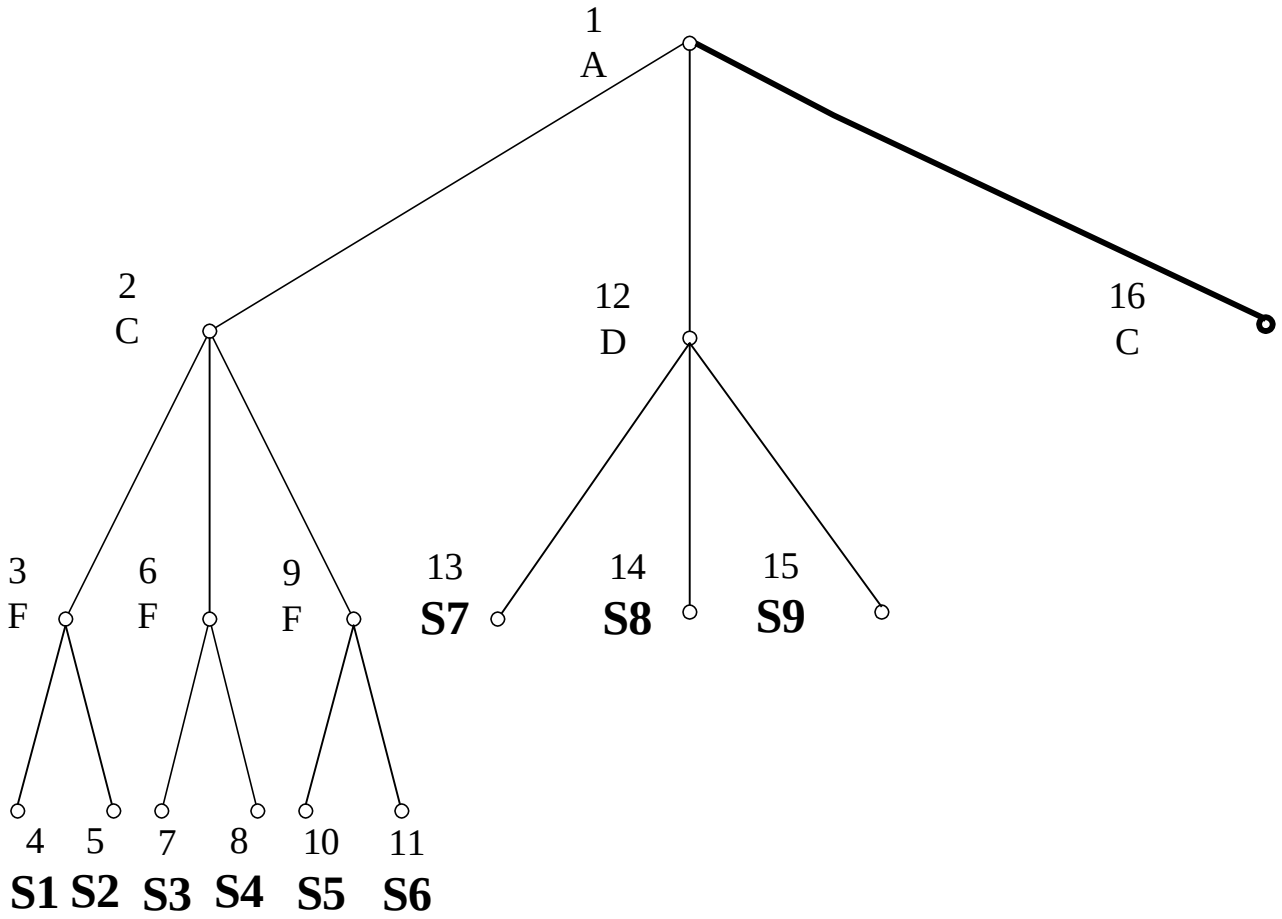
Partial problem #12





Partial problem #15 (S9)

Enumeration (search) tree (worst case)

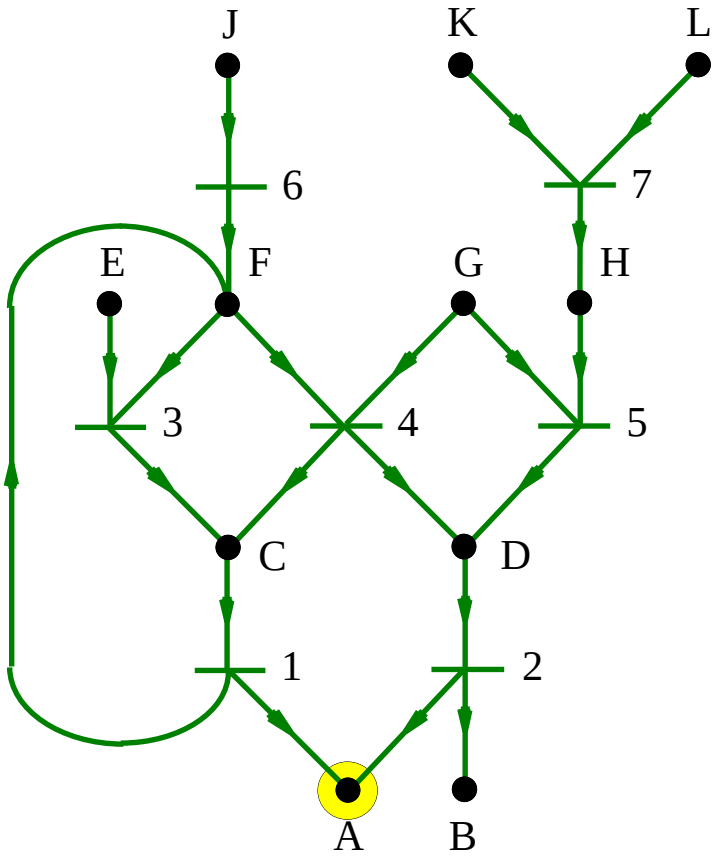


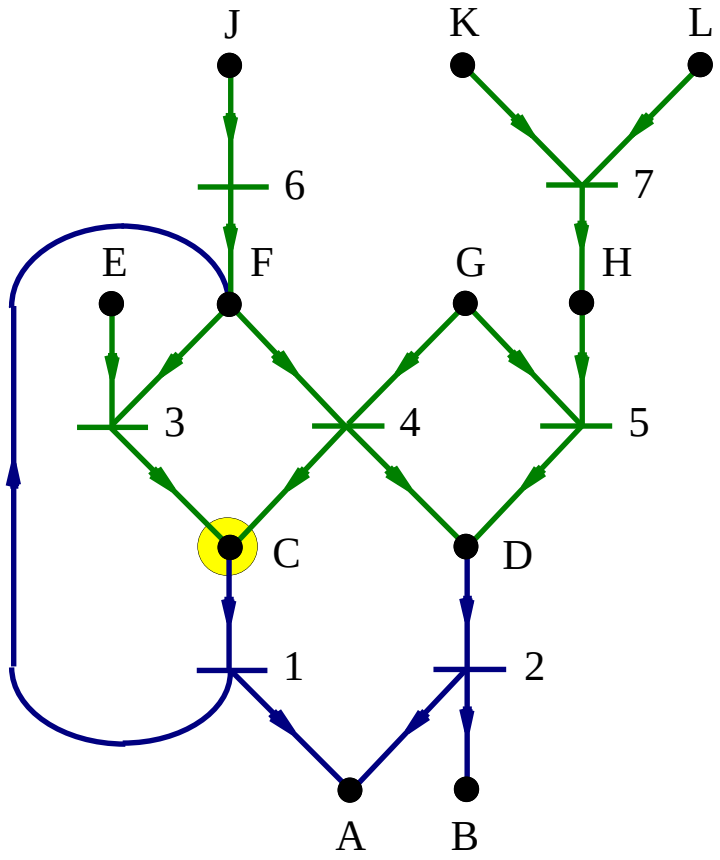
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

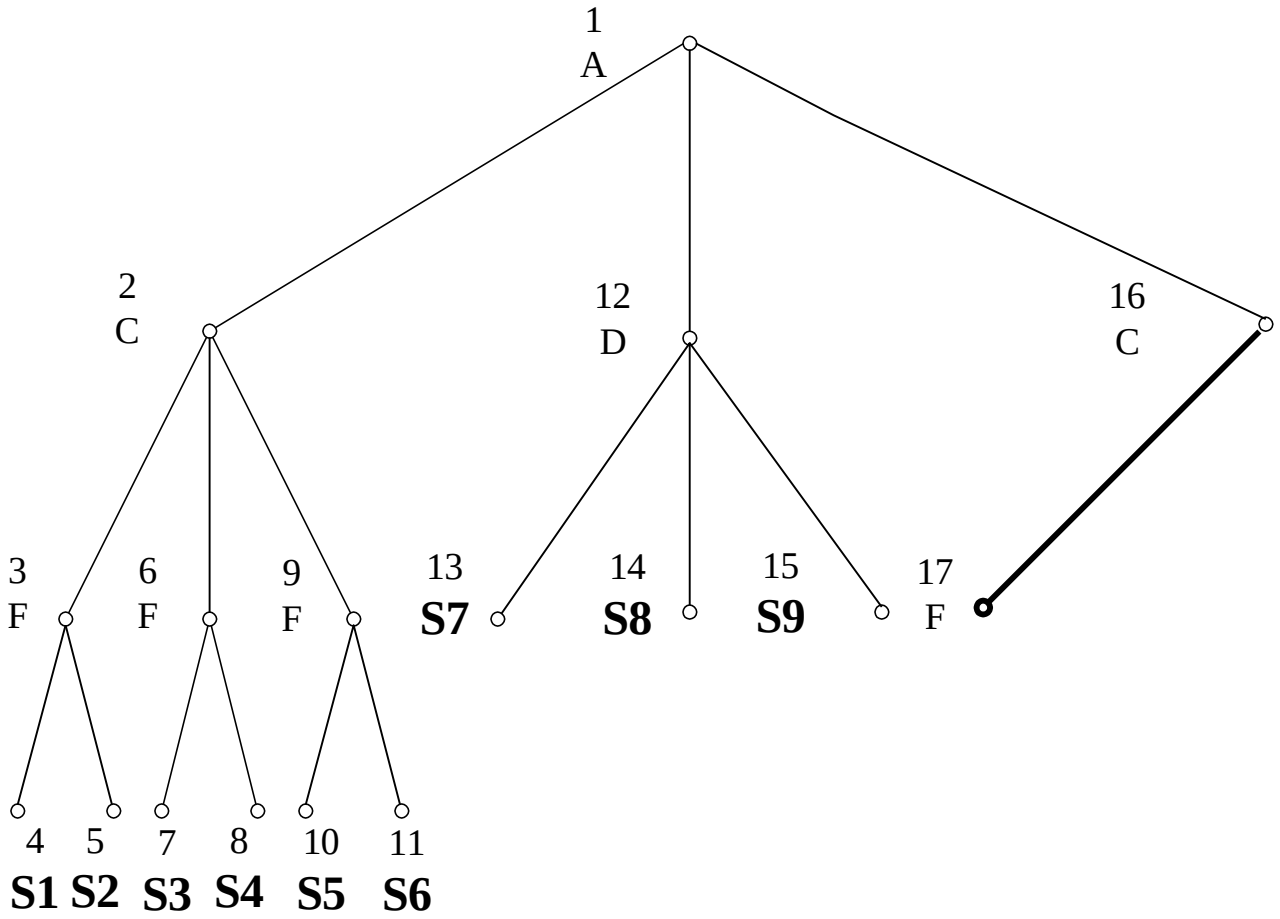
Partial problem #1





Partial problem #16

Enumeration (search) tree (worst case)

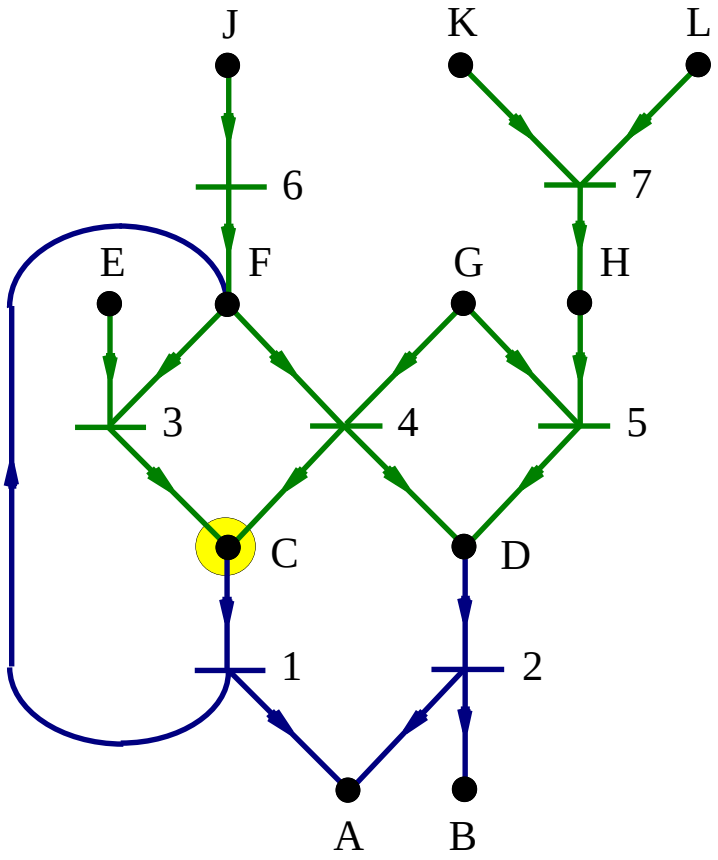


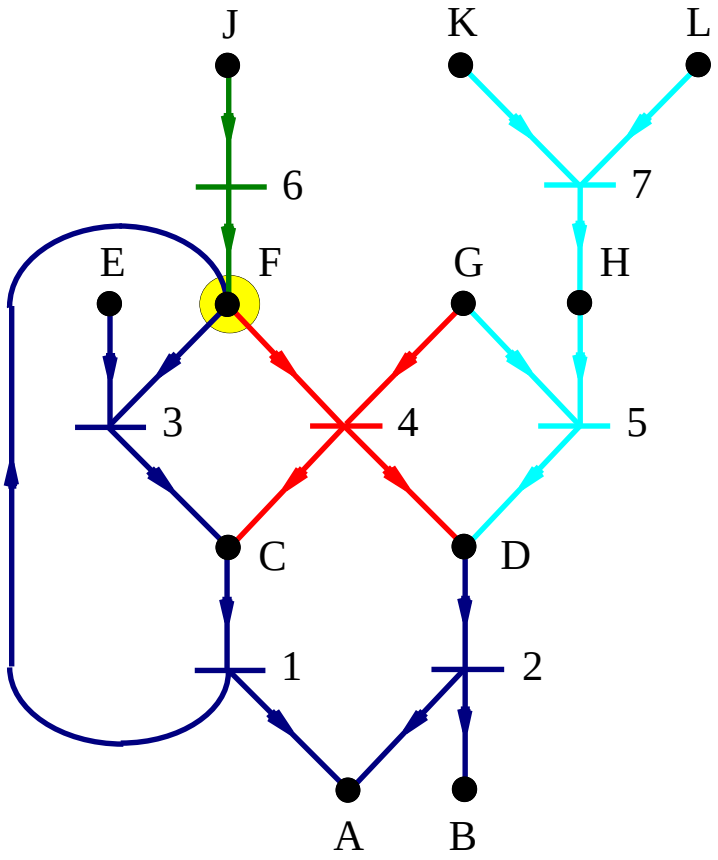
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

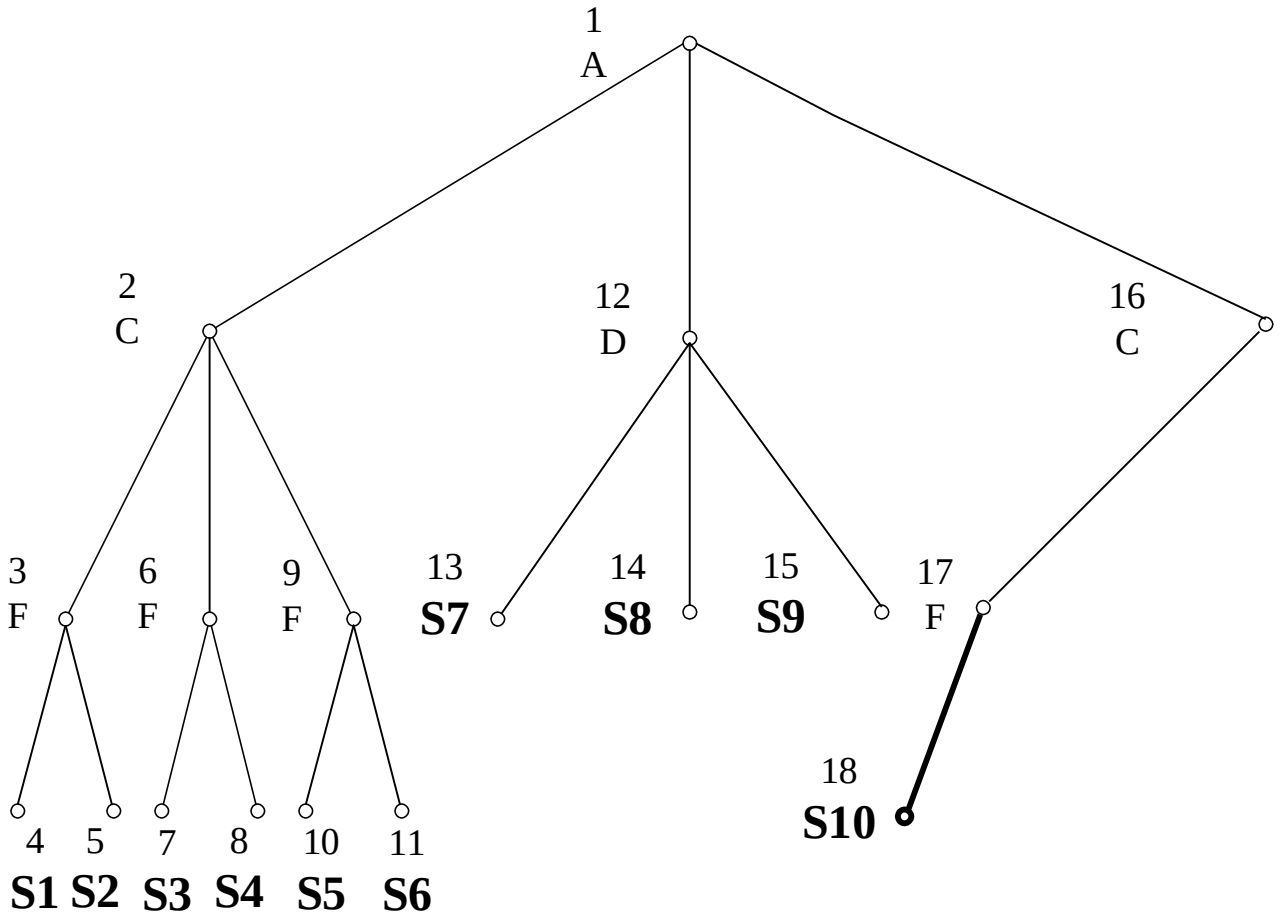
Partial problem #16





Partial problem #17

Enumeration (search) tree (worst case)

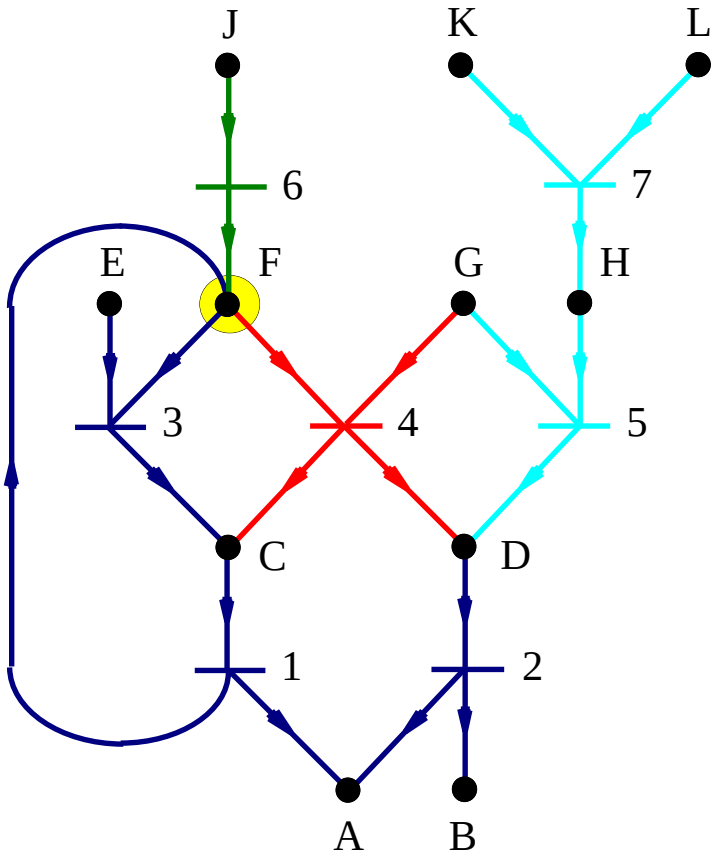


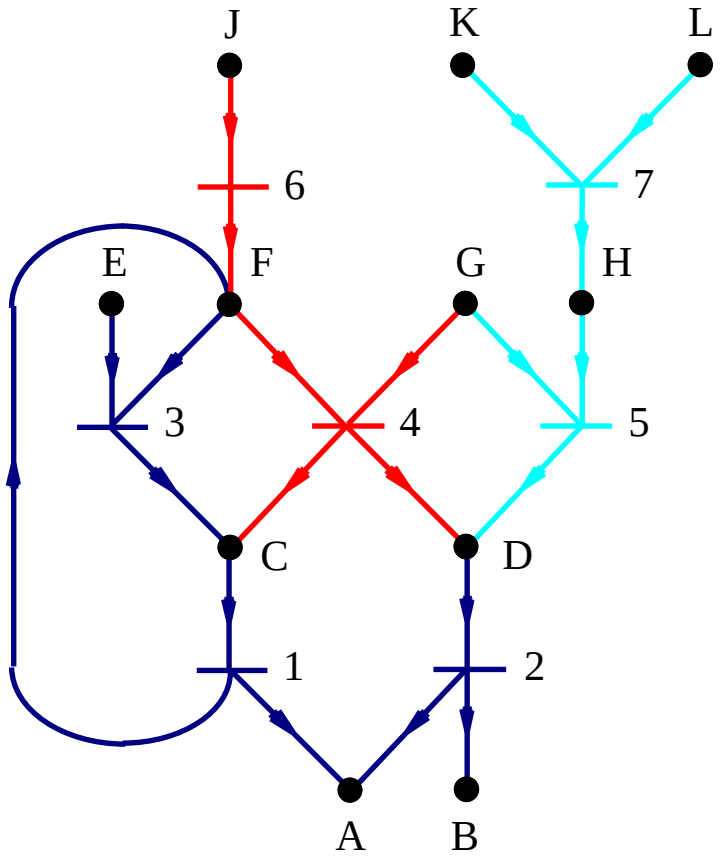
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

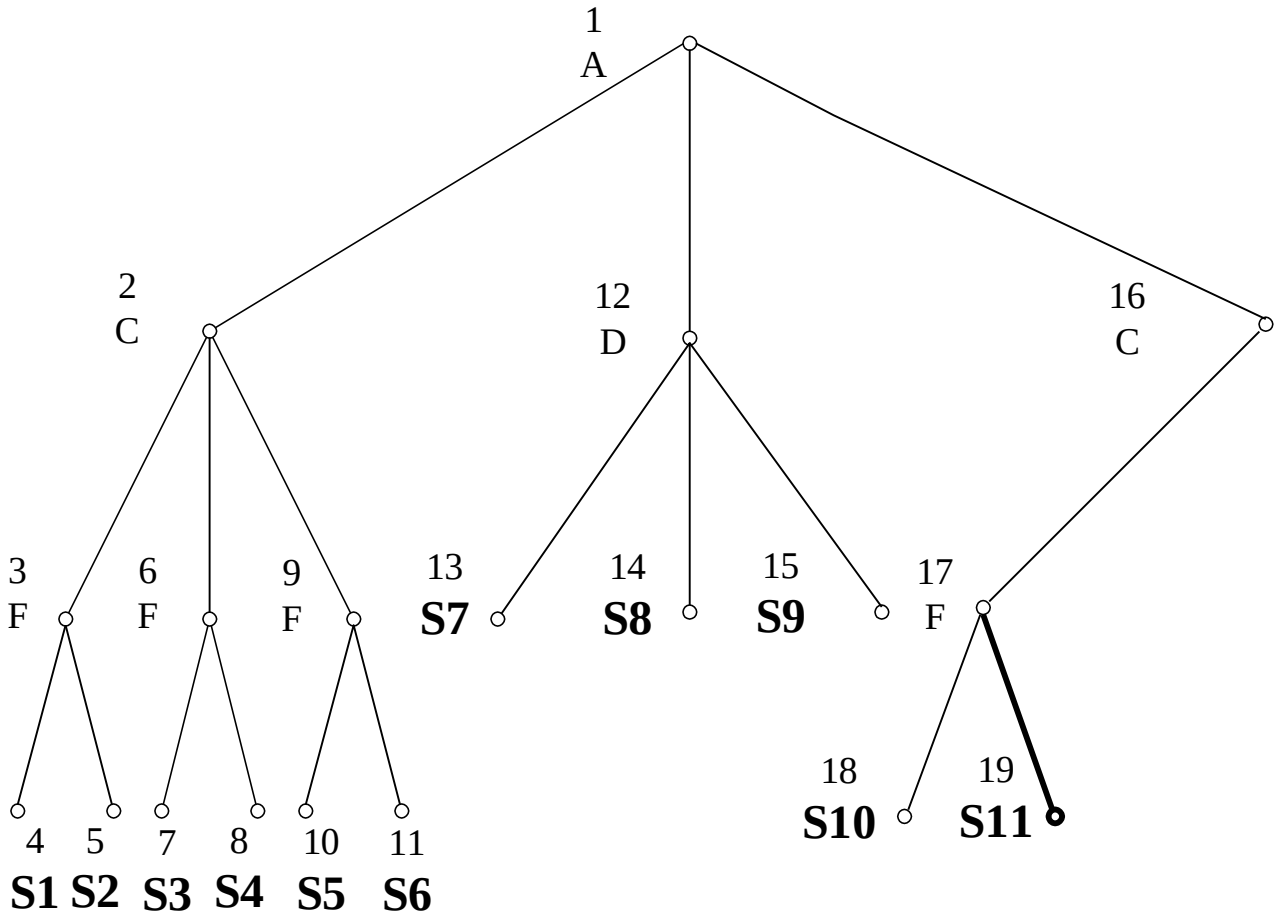
Partial problem #17





Partial problem #18 (S10)

Enumeration (search) tree (worst case)

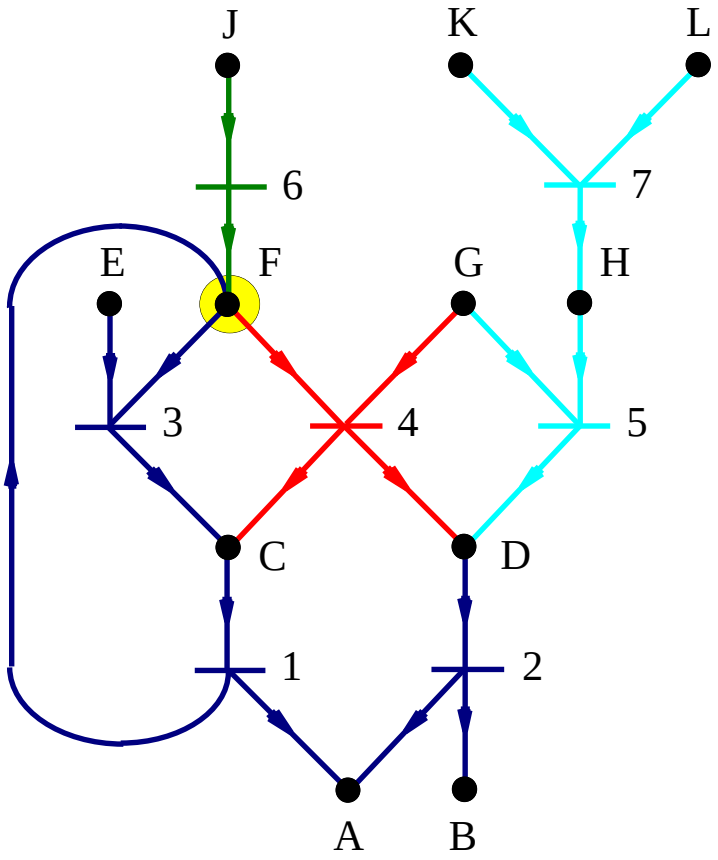


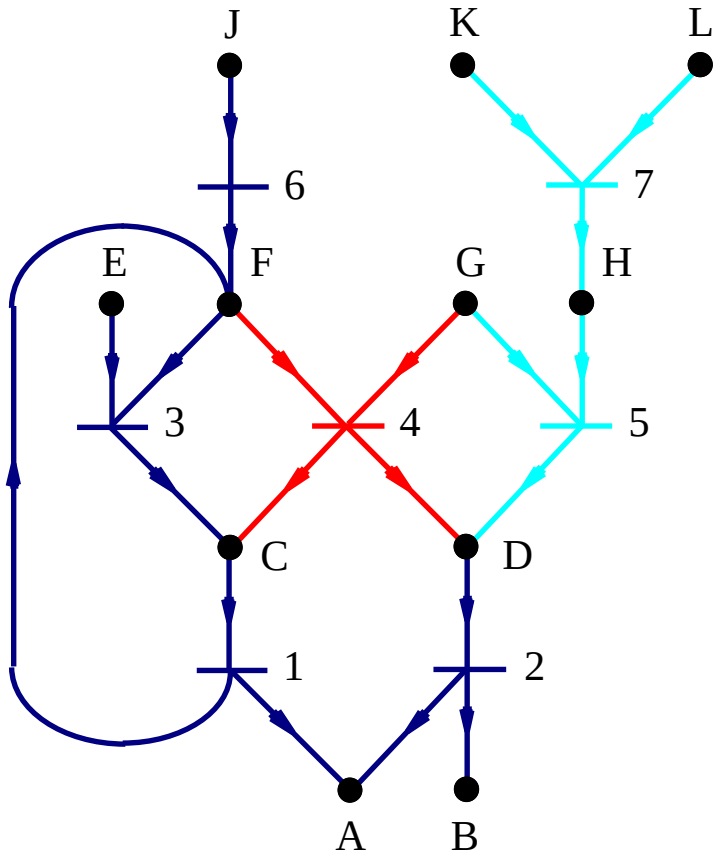
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

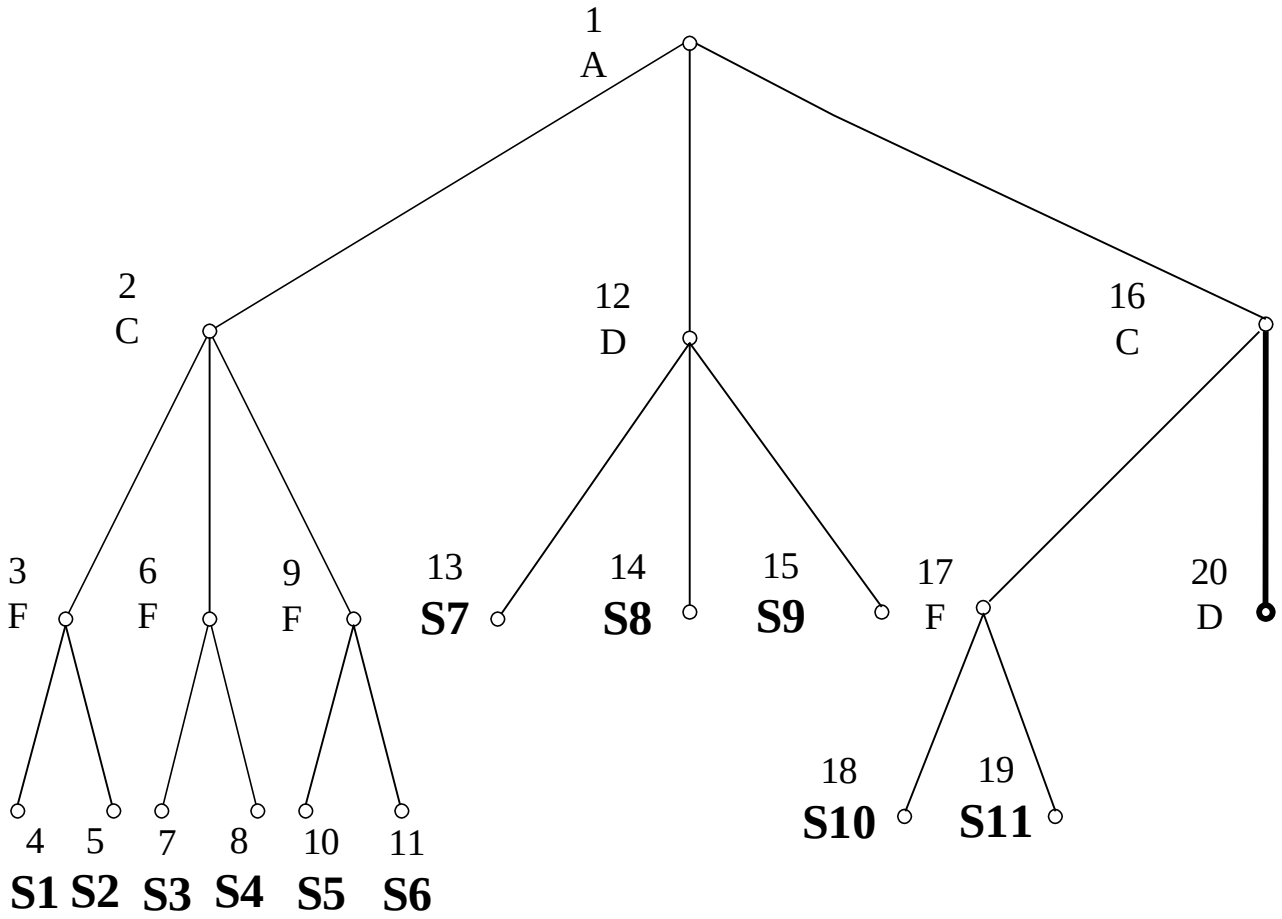
Partial problem #17





Partial problem #19 (S11)

Enumeration (search) tree (worst case)

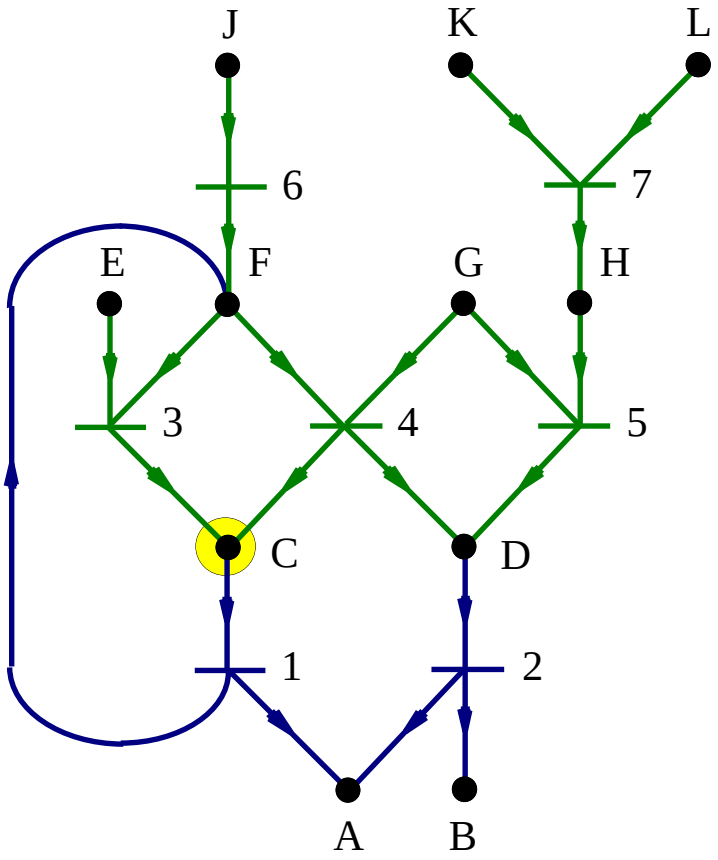


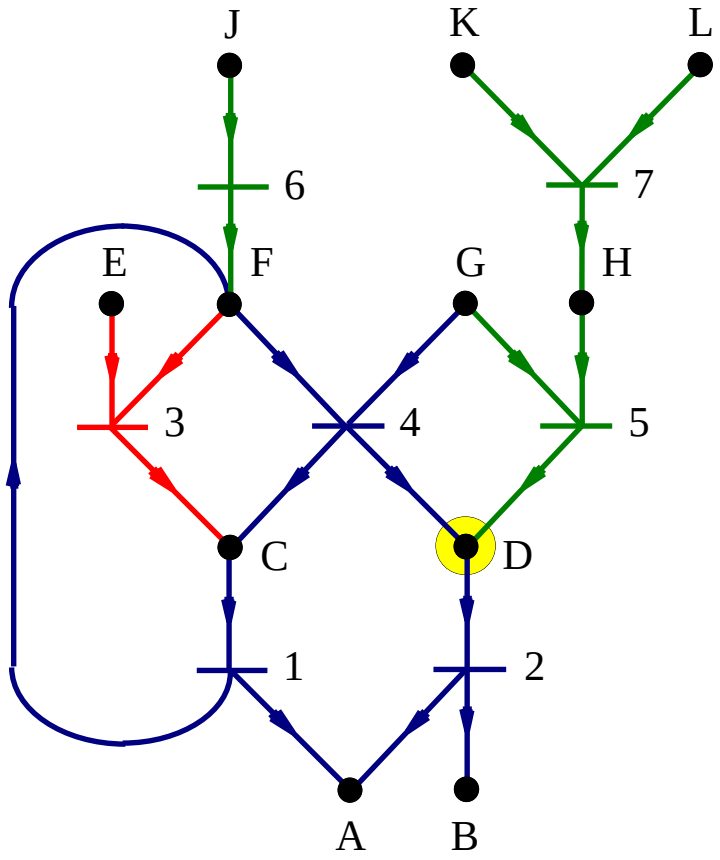
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

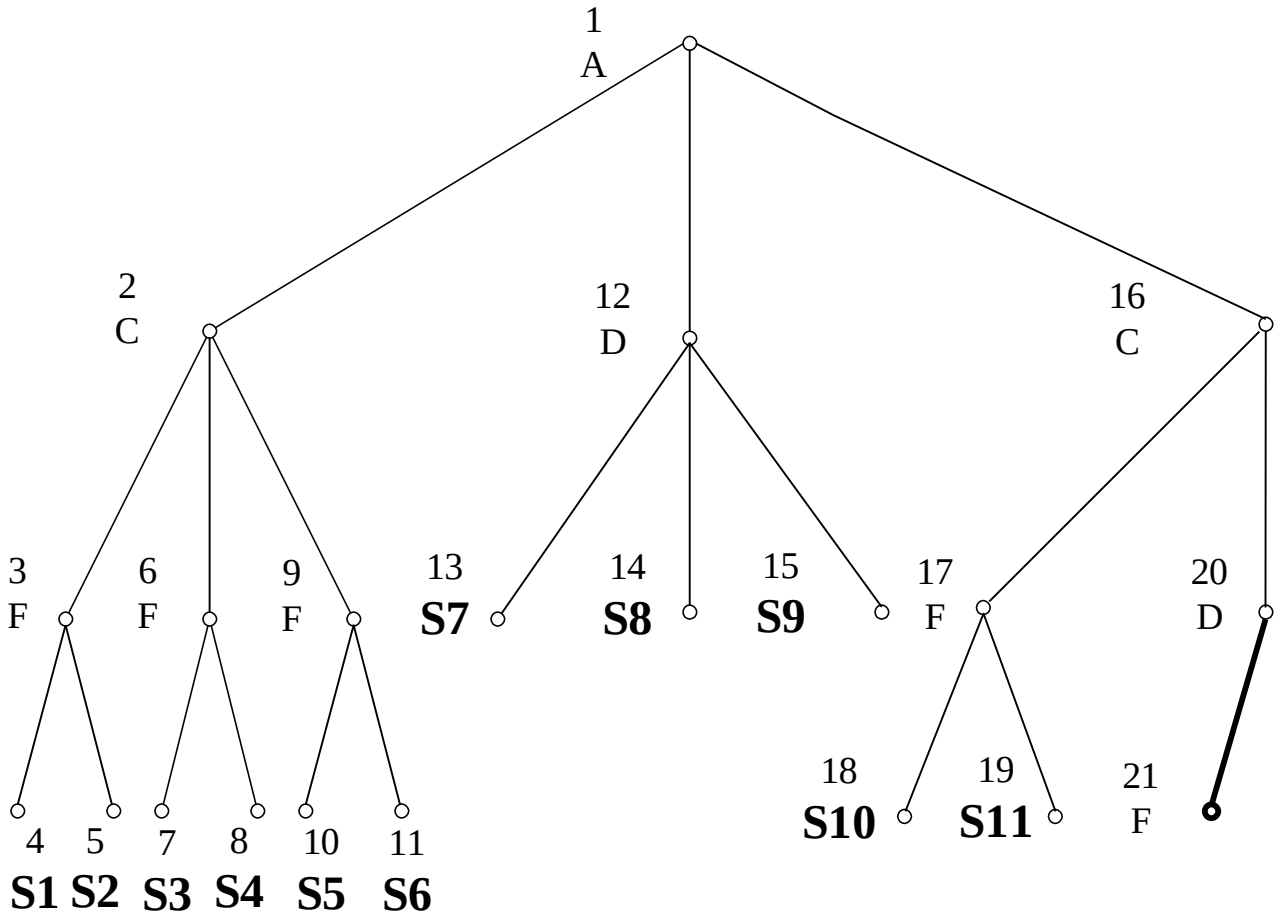
Partial problem #16





Partial problem #20

Enumeration (search) tree (worst case)

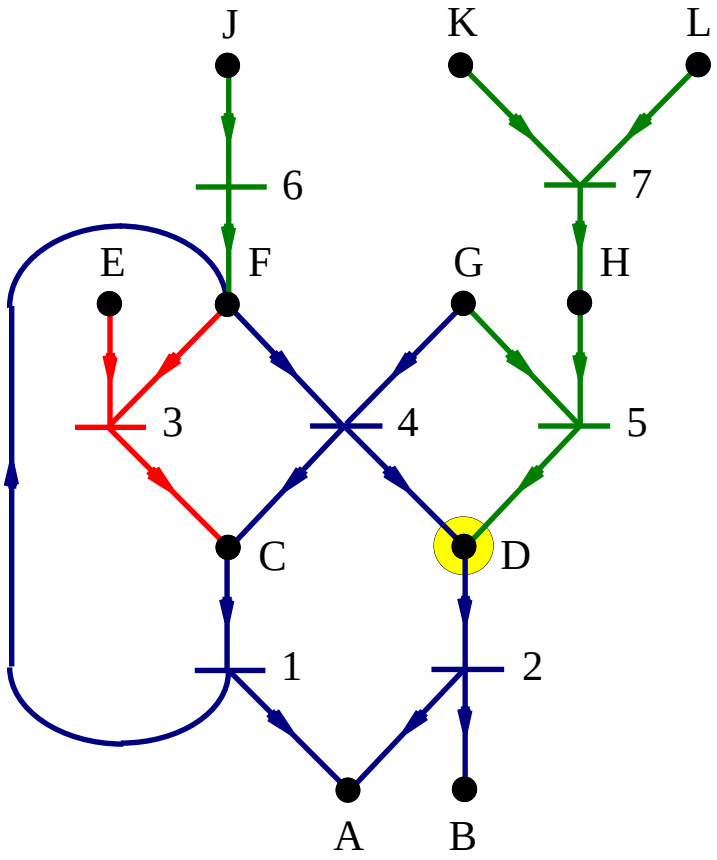


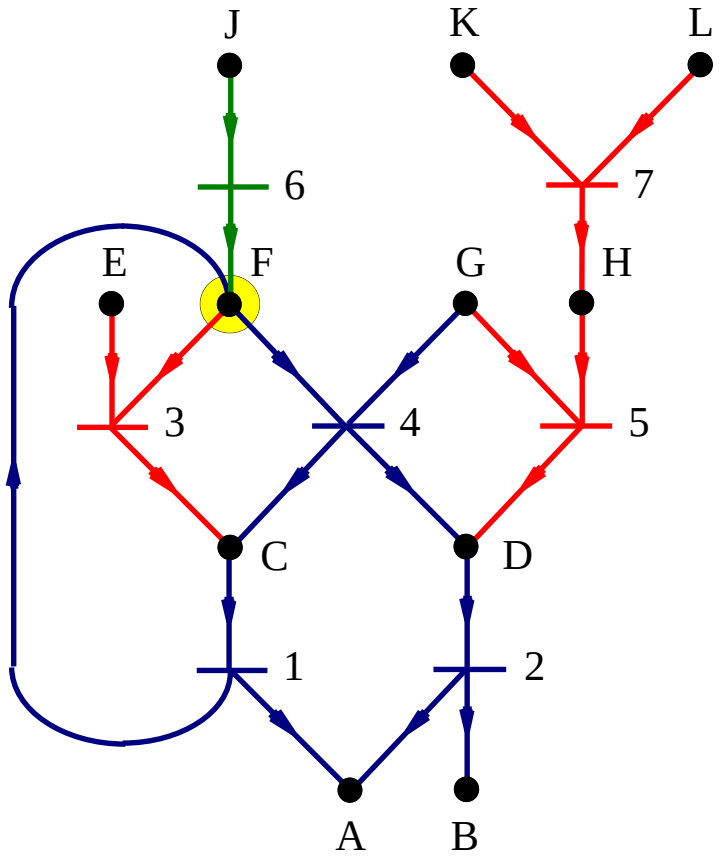
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

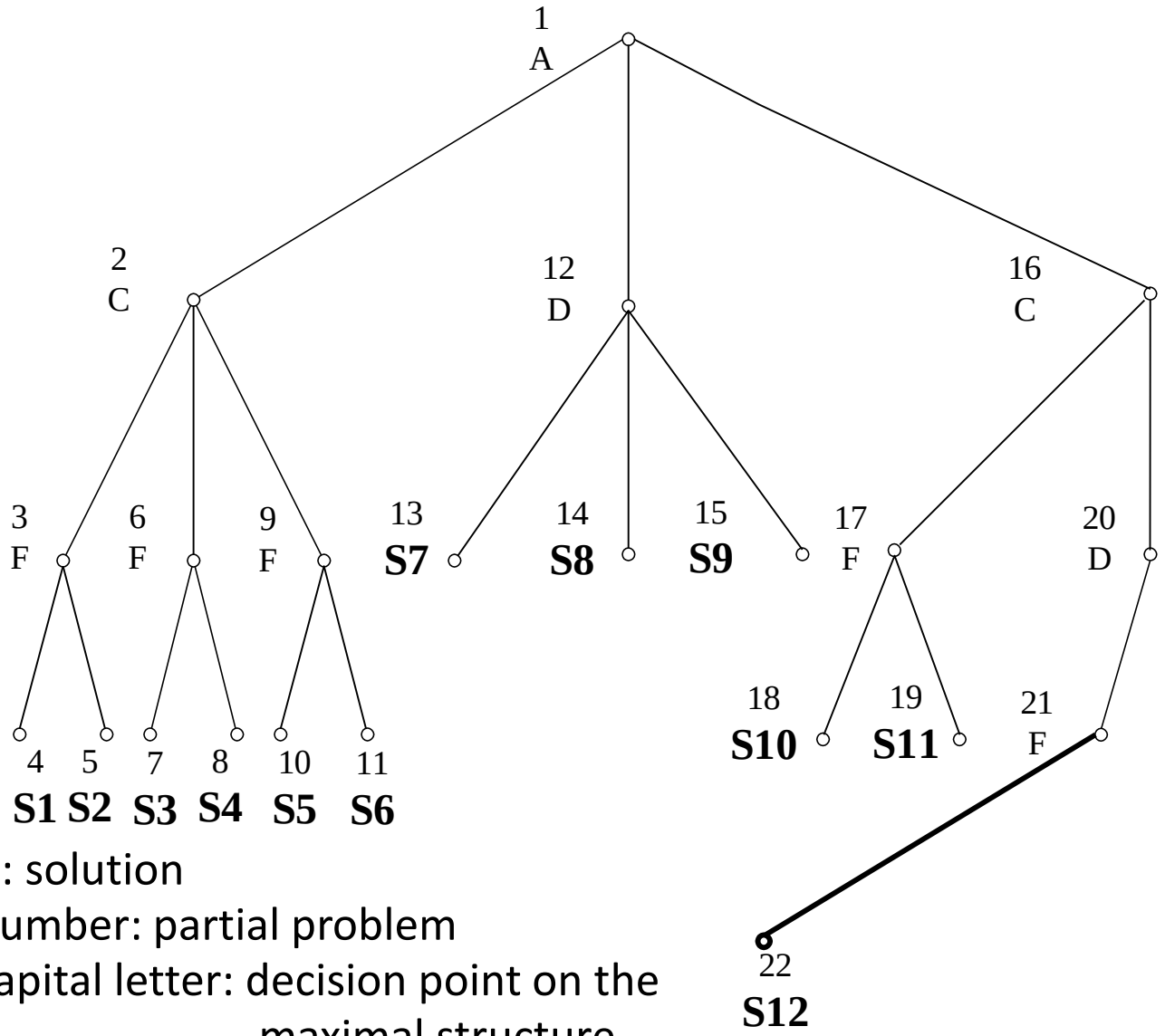
Partial problem #20





Partial problem #21

Enumeration (search) tree (worst case)

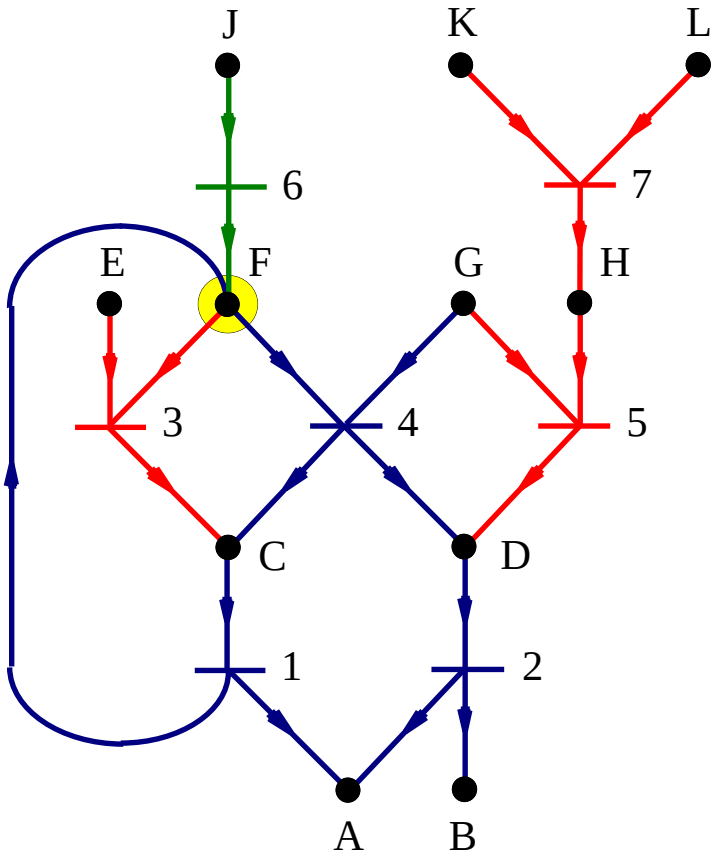


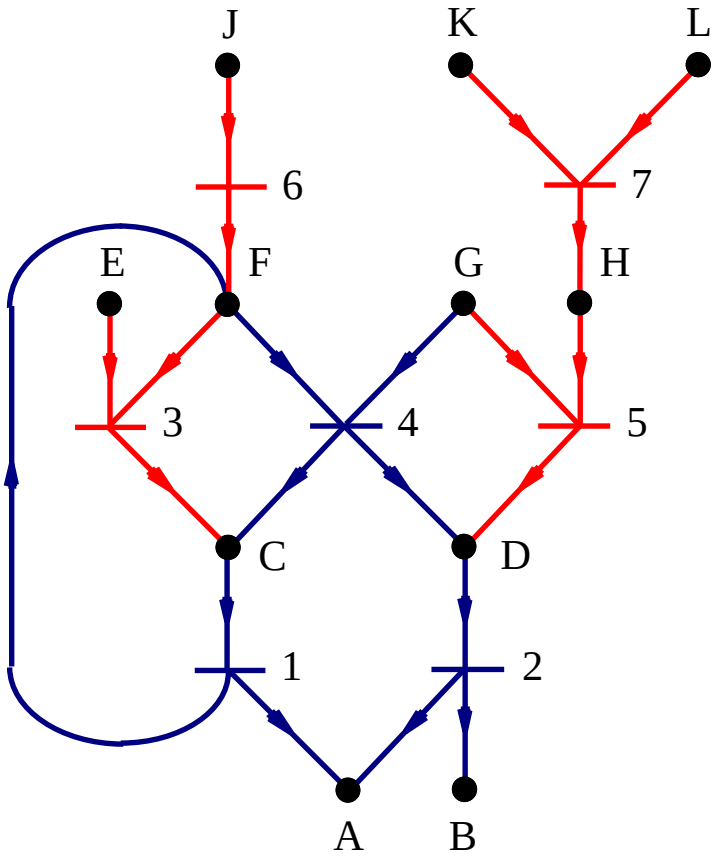
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

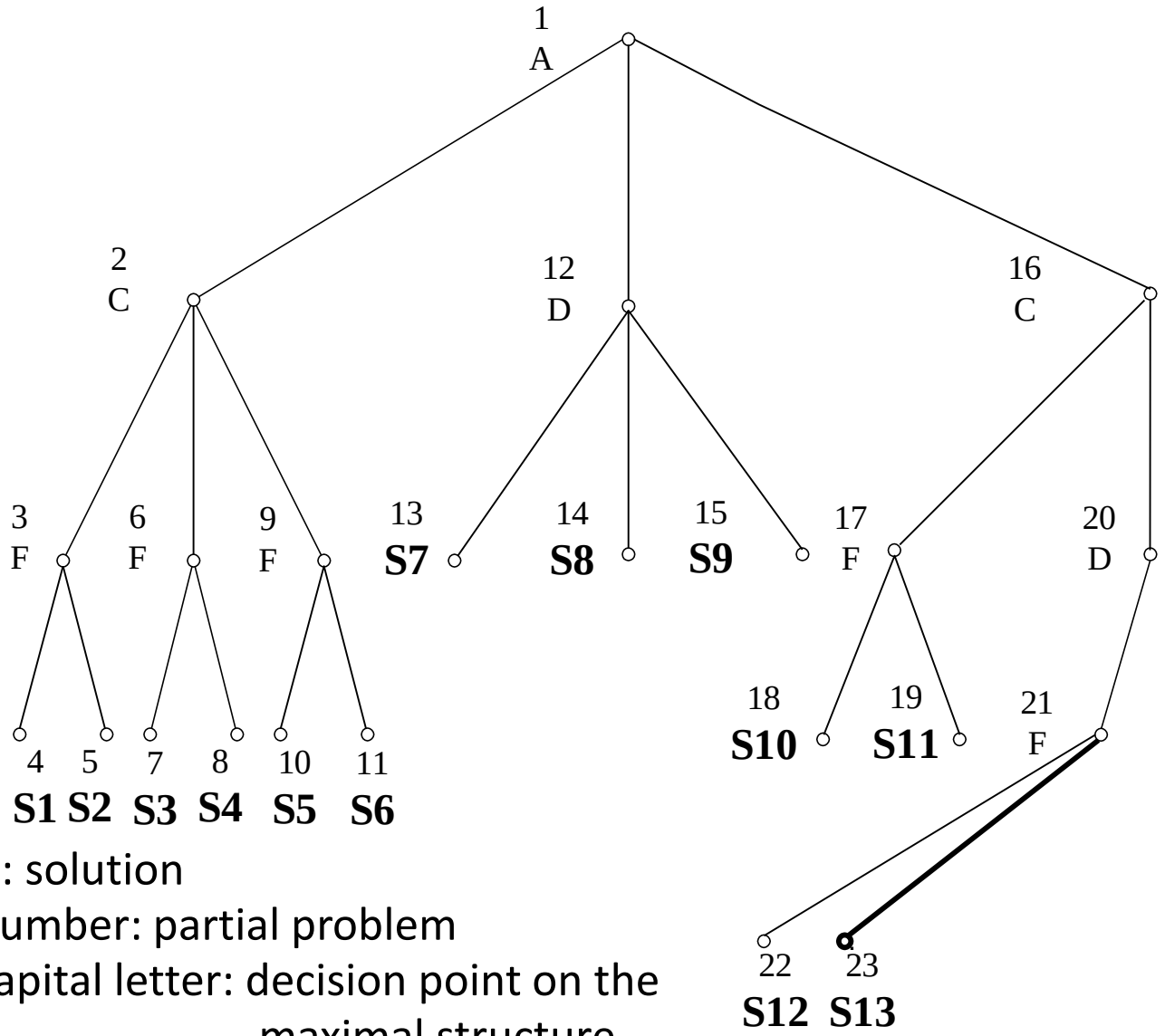
Partial problem #21





Partial problem #22 (S12)

Enumeration (search) tree (worst case)

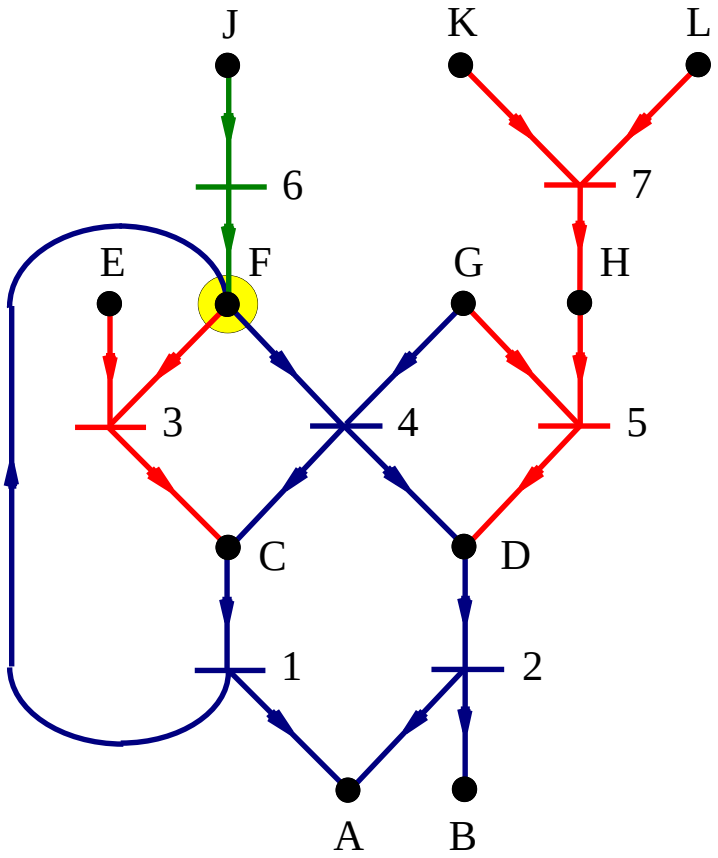


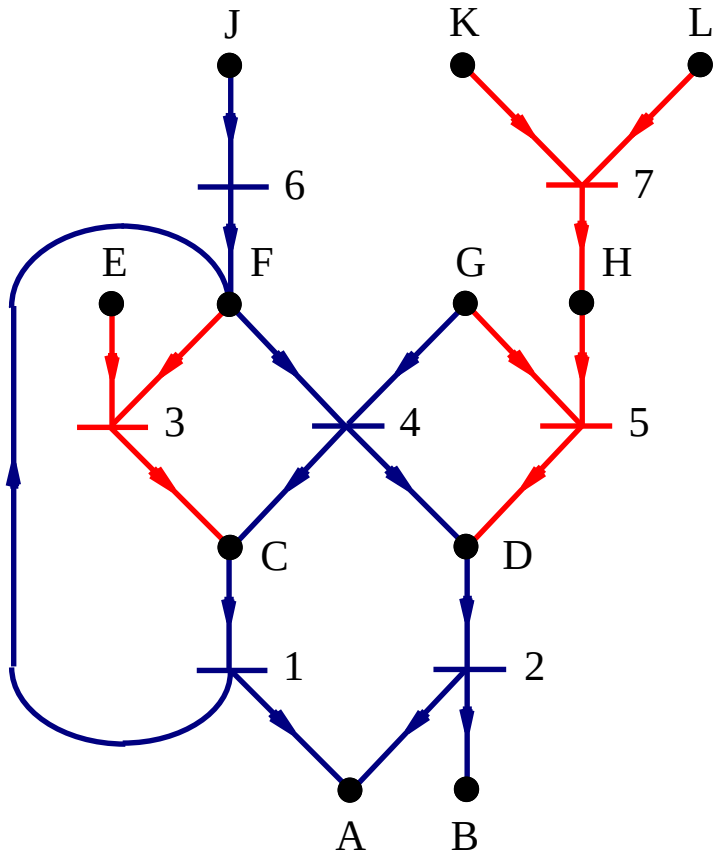
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

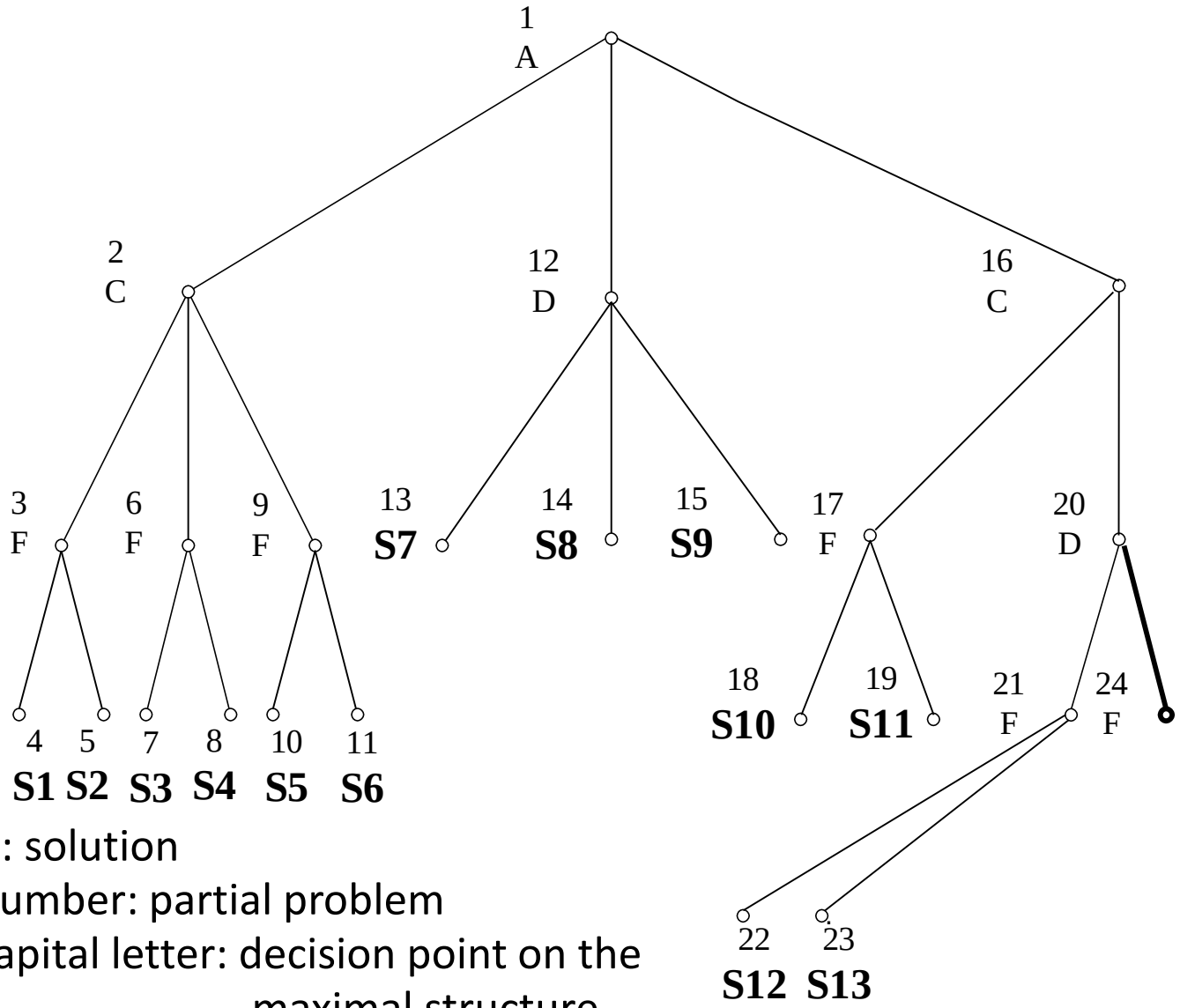
Partial problem #21





Partial problem #23 (S13)

Enumeration (search) tree (worst case)

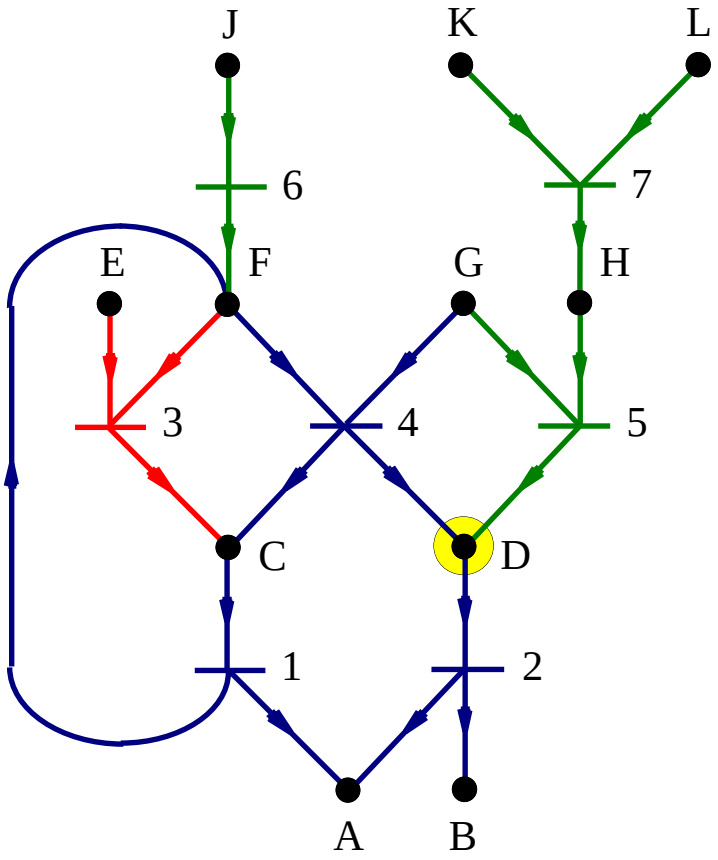


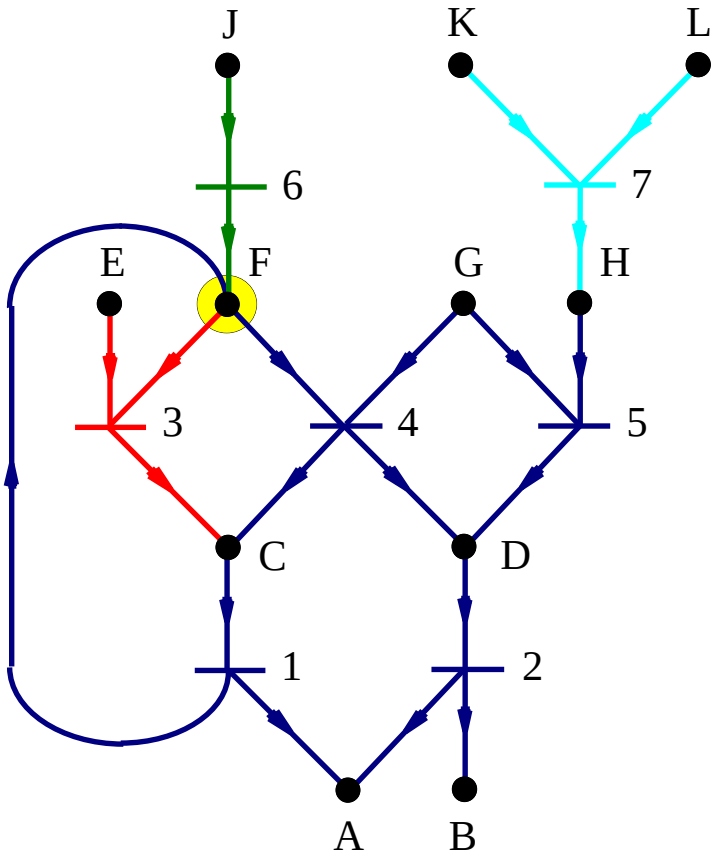
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

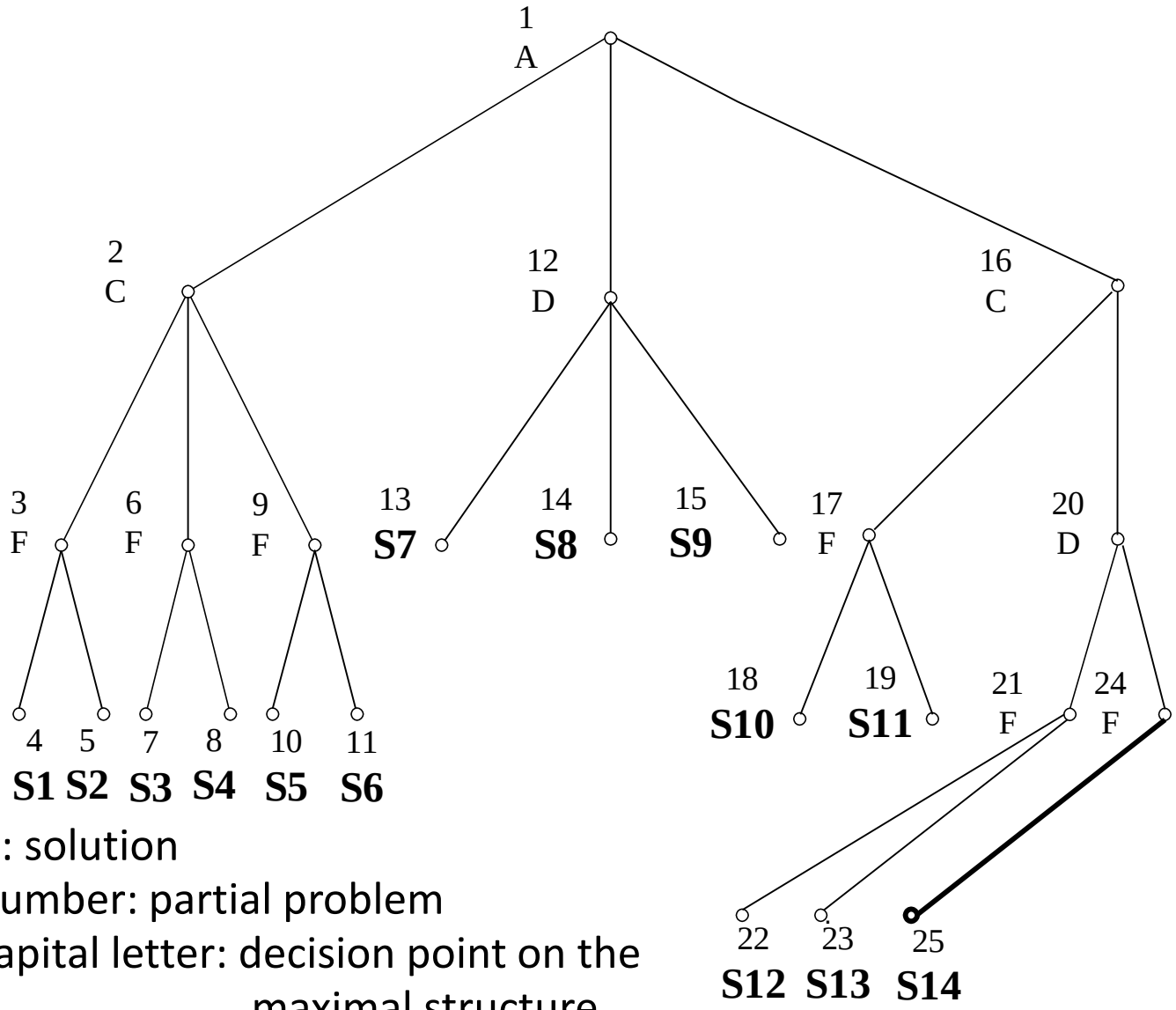
Partial problem #20





Partial problem #24

Enumeration (search) tree (worst case)

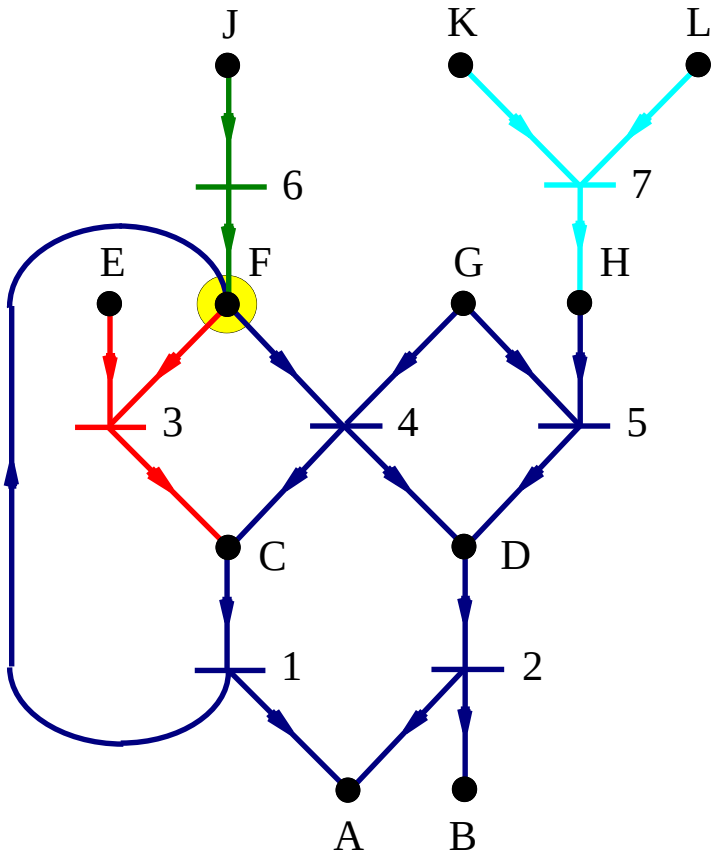


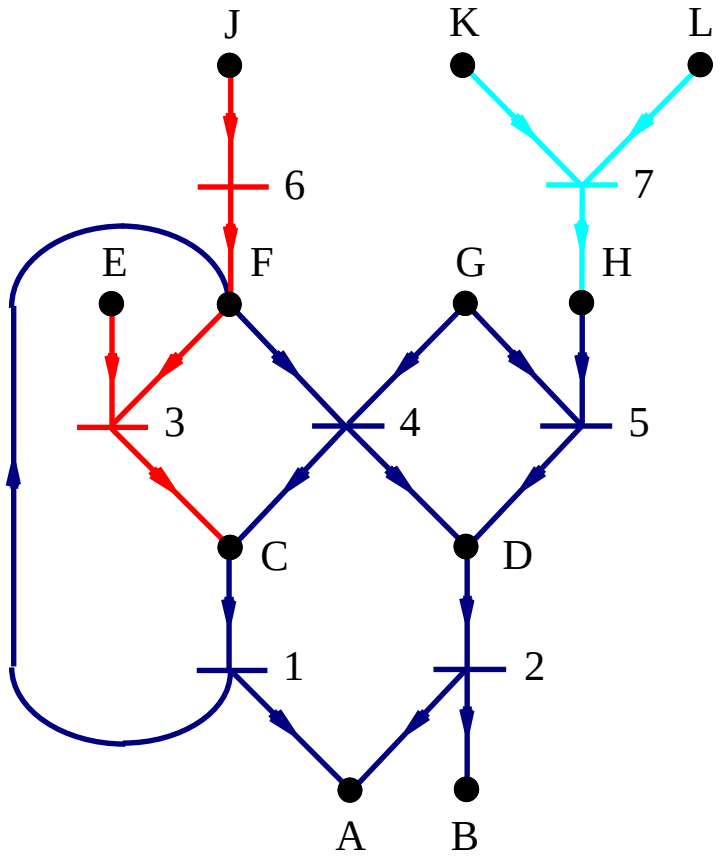
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

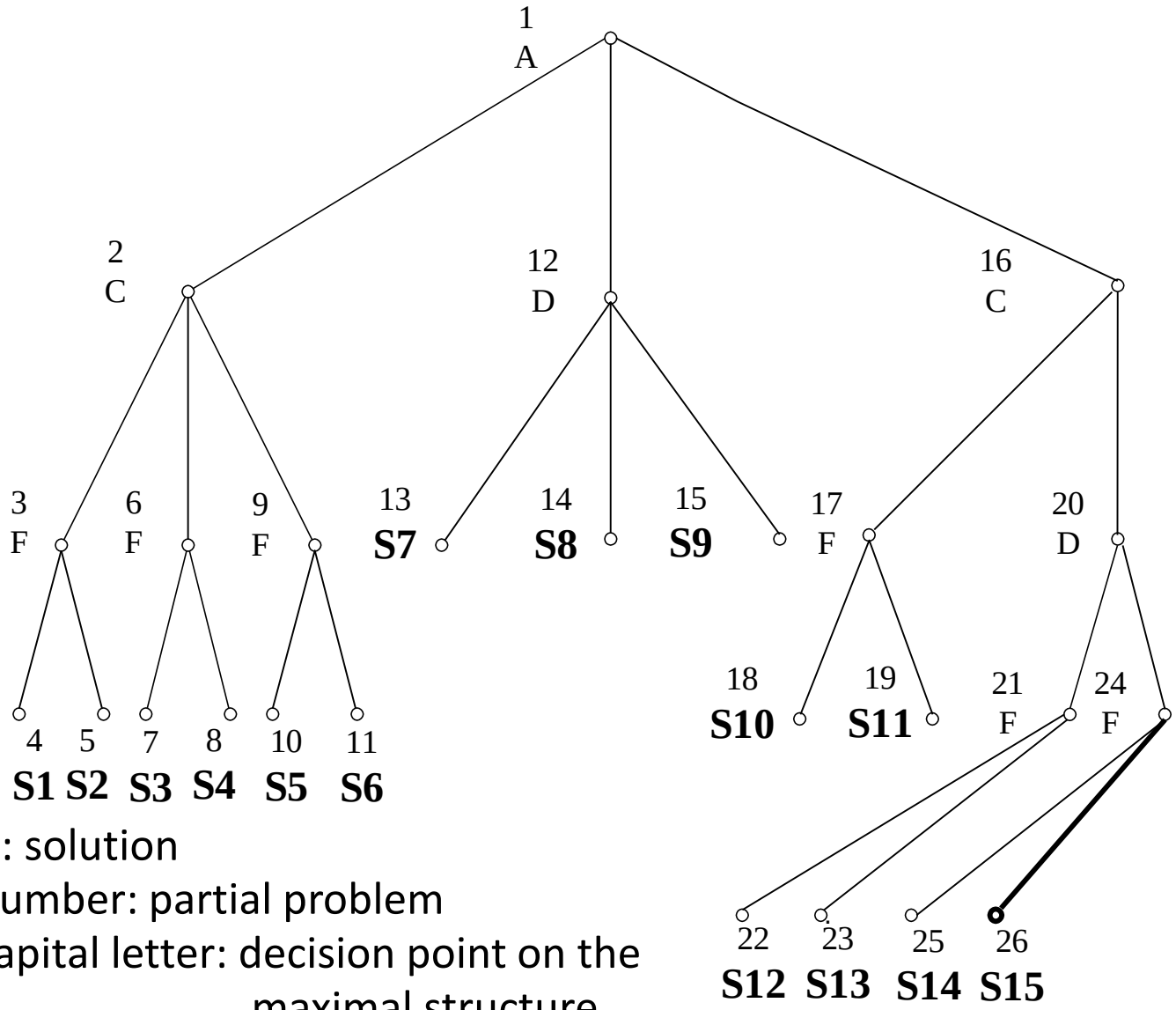
Partial problem #24





Partial problem #25 (S14)

Enumeration (search) tree (worst case)

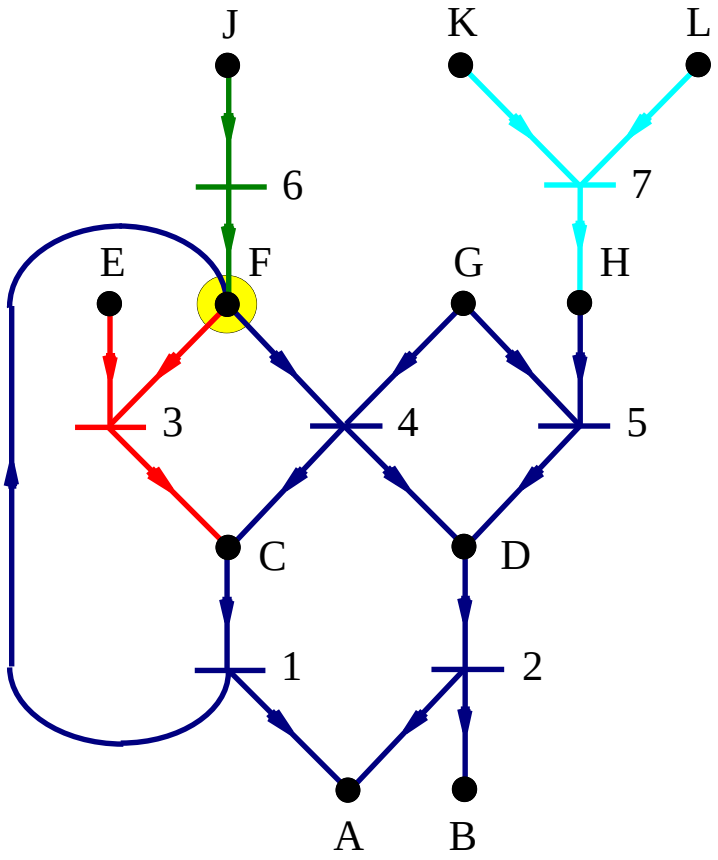


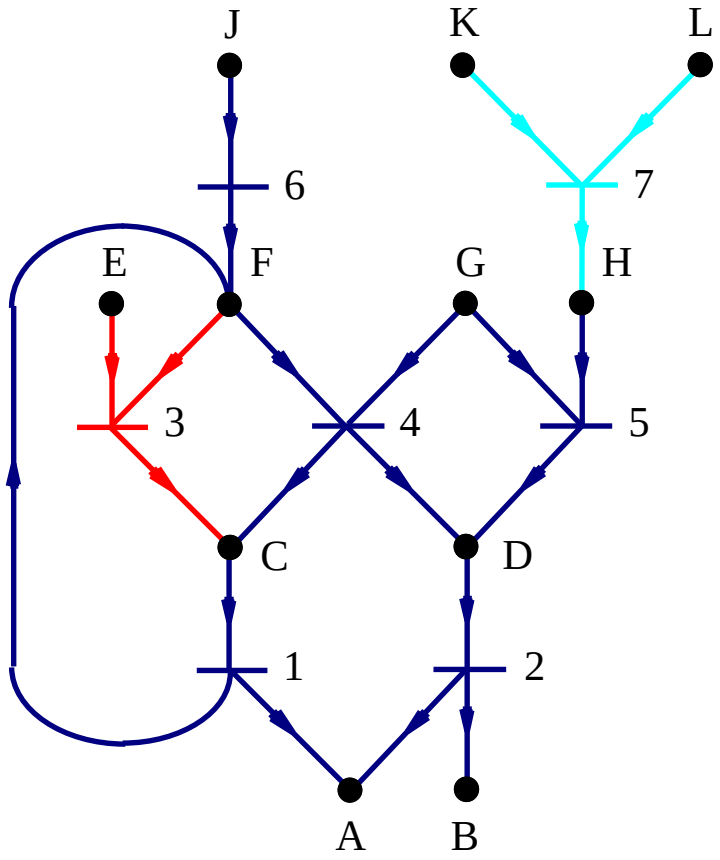
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

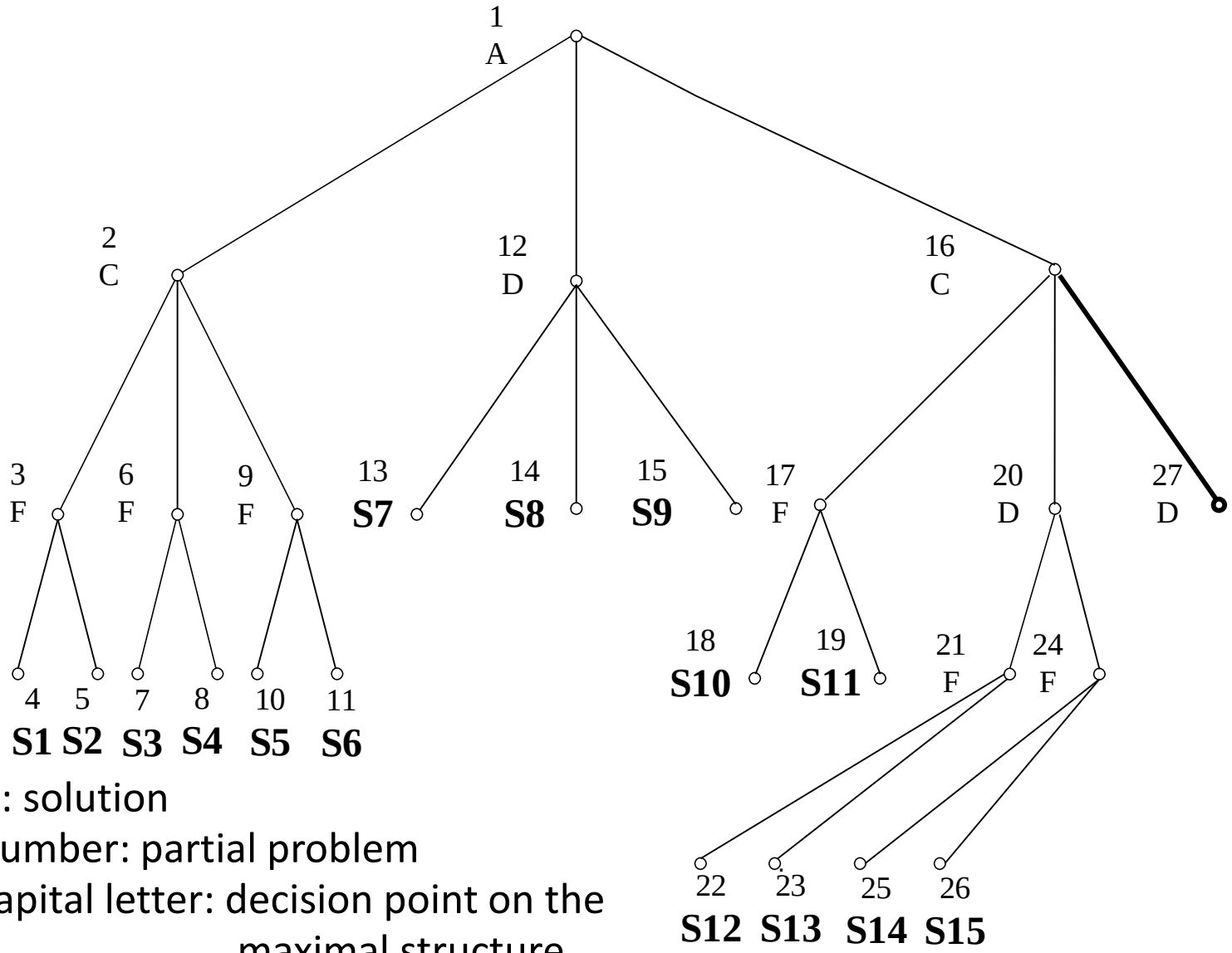
Partial problem #24





Partial problem #26 (S15)

Enumeration (search) tree (worst case)

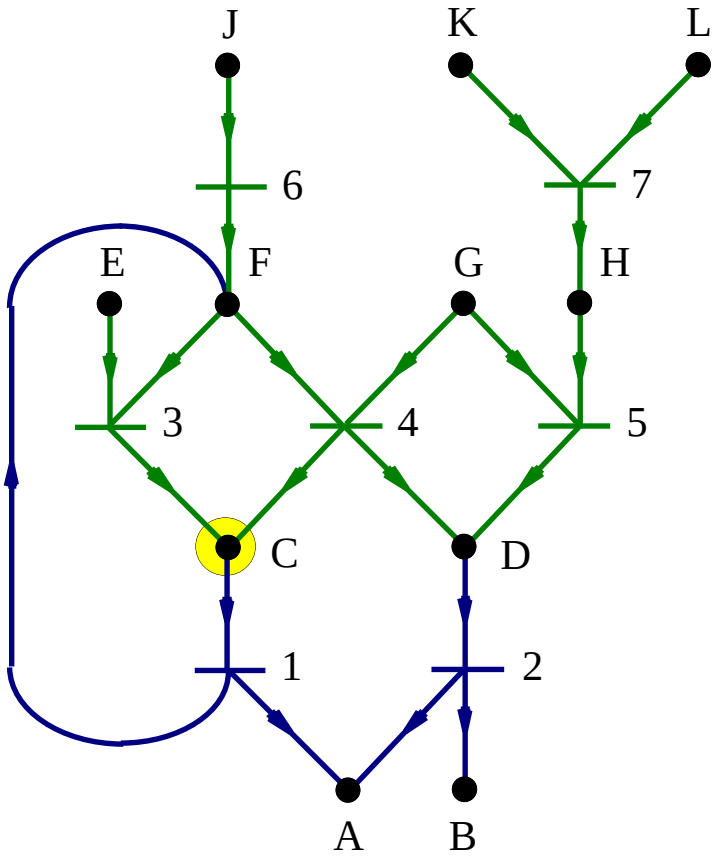


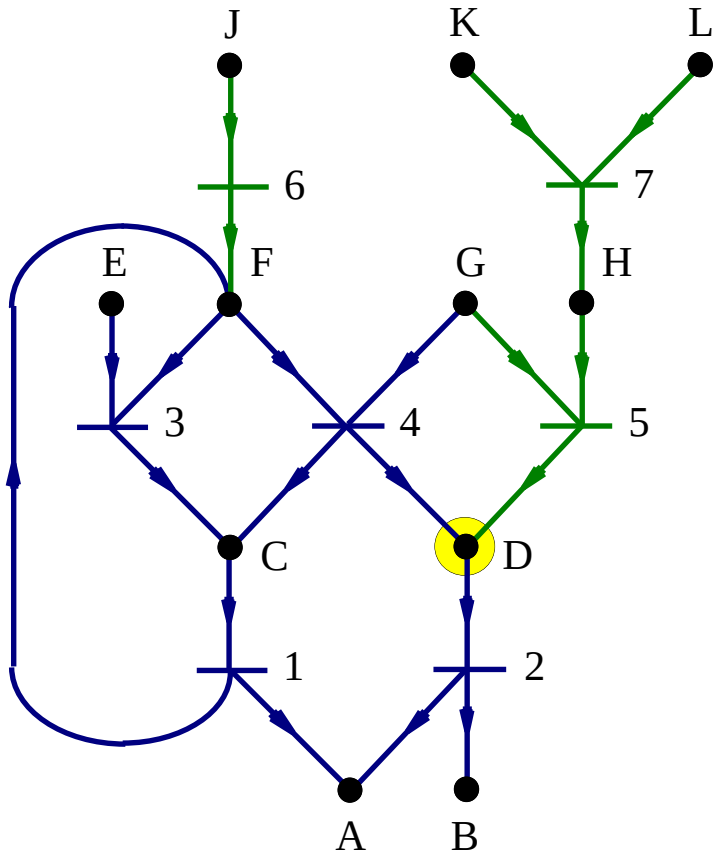
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

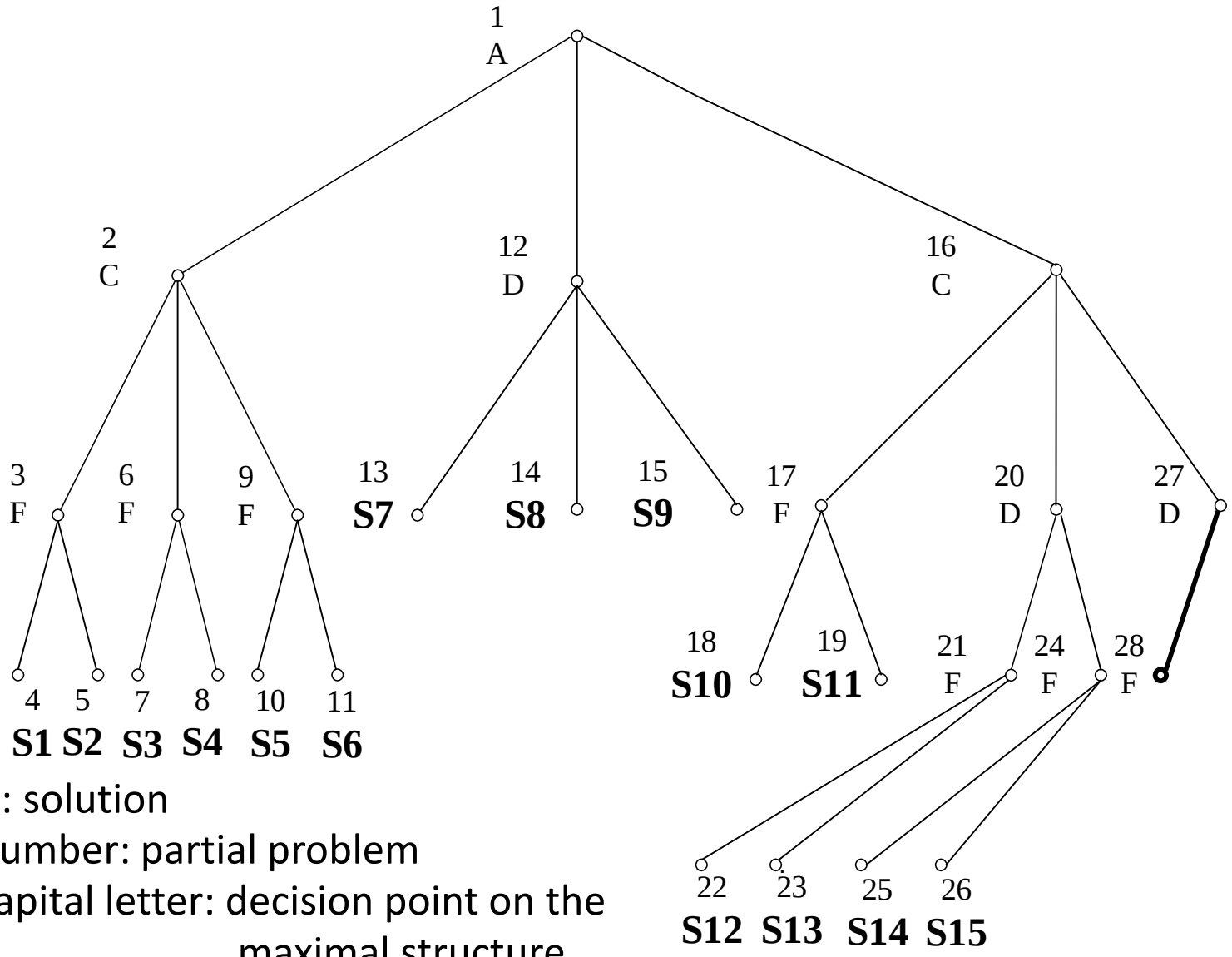
Partial problem #16





Partial problem #27

Enumeration (search) tree (worst case)

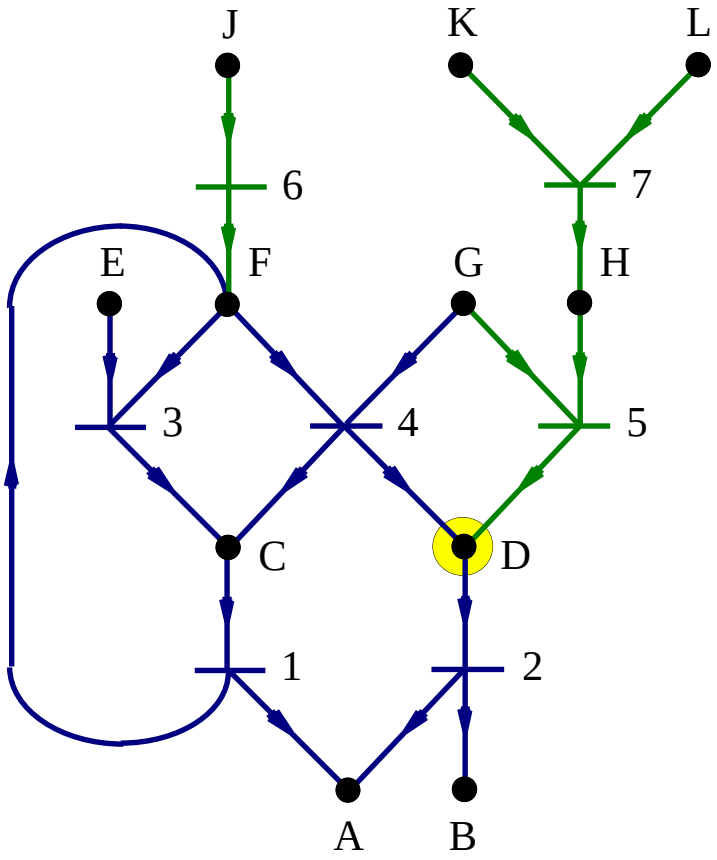


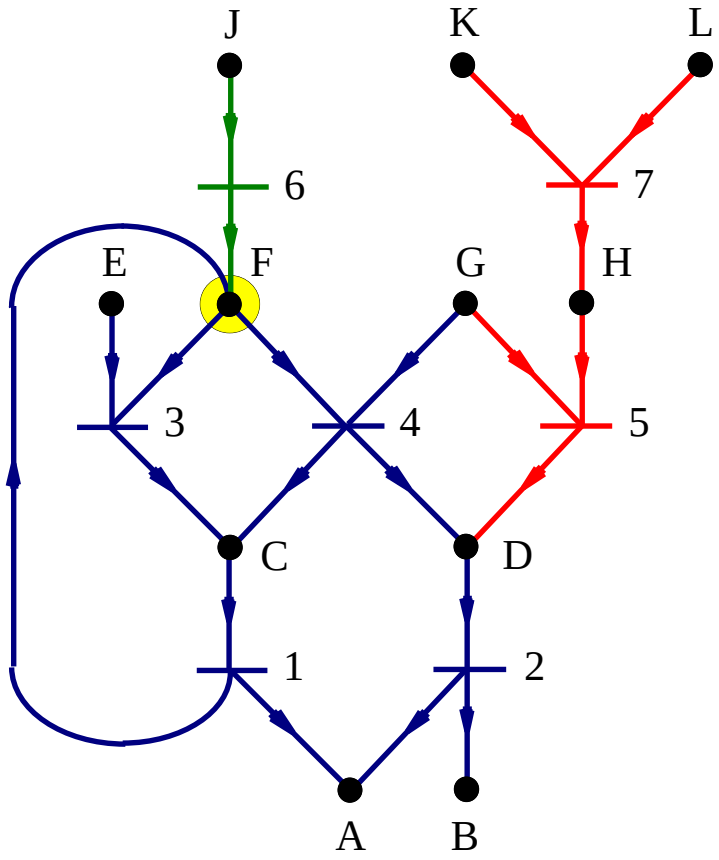
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

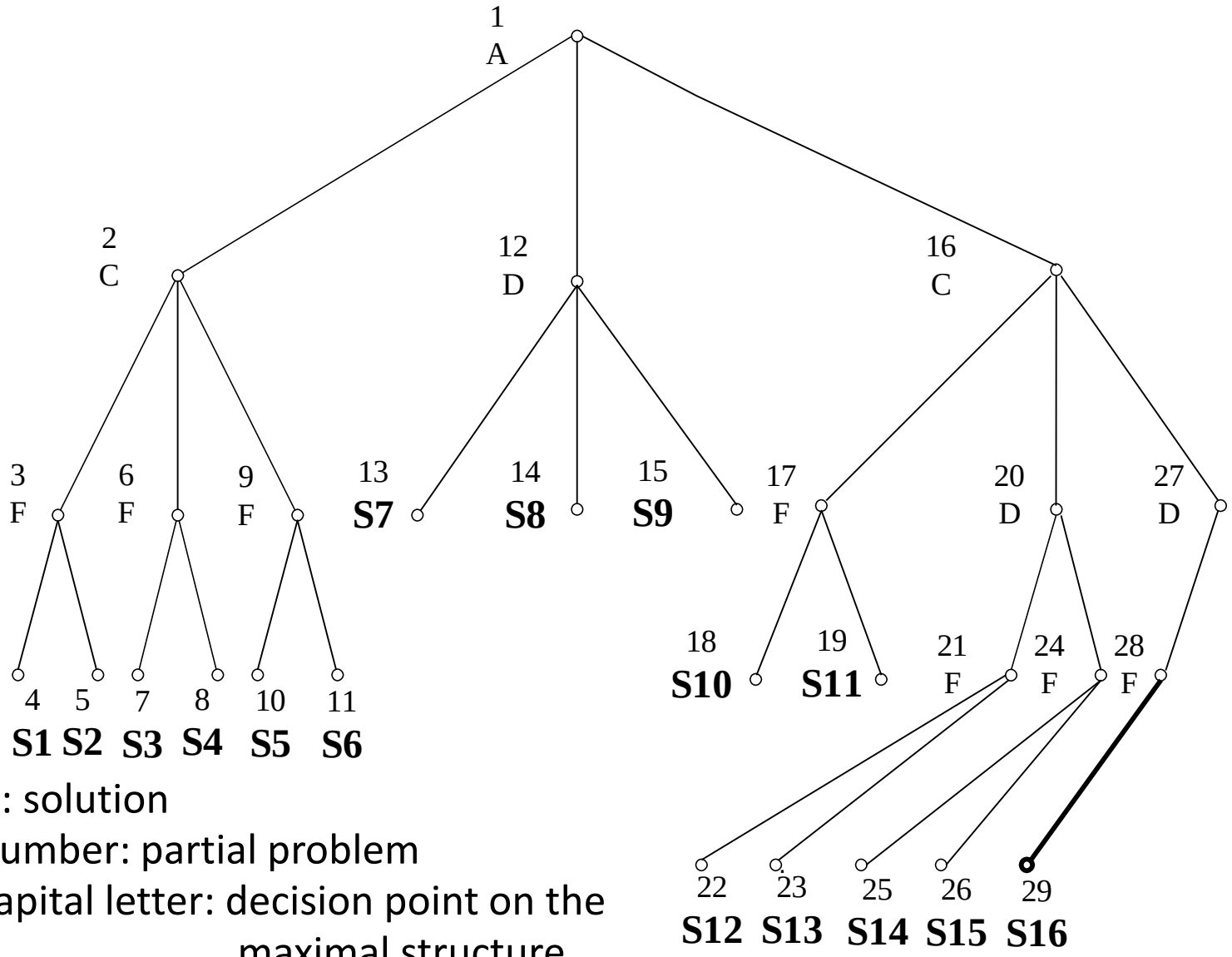
Partial problem #27





Partial problem #28

Enumeration (search) tree (worst case)

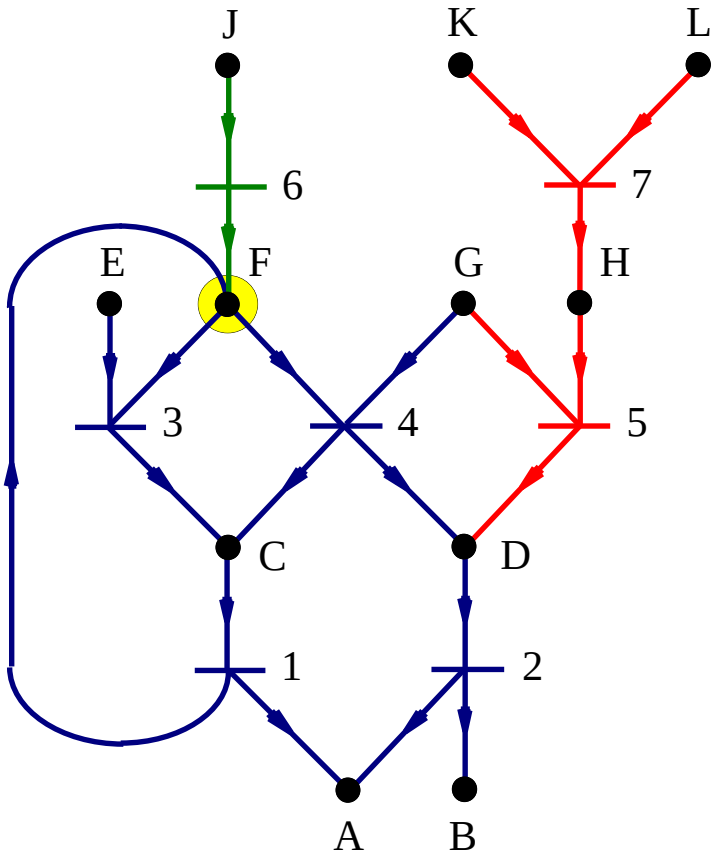


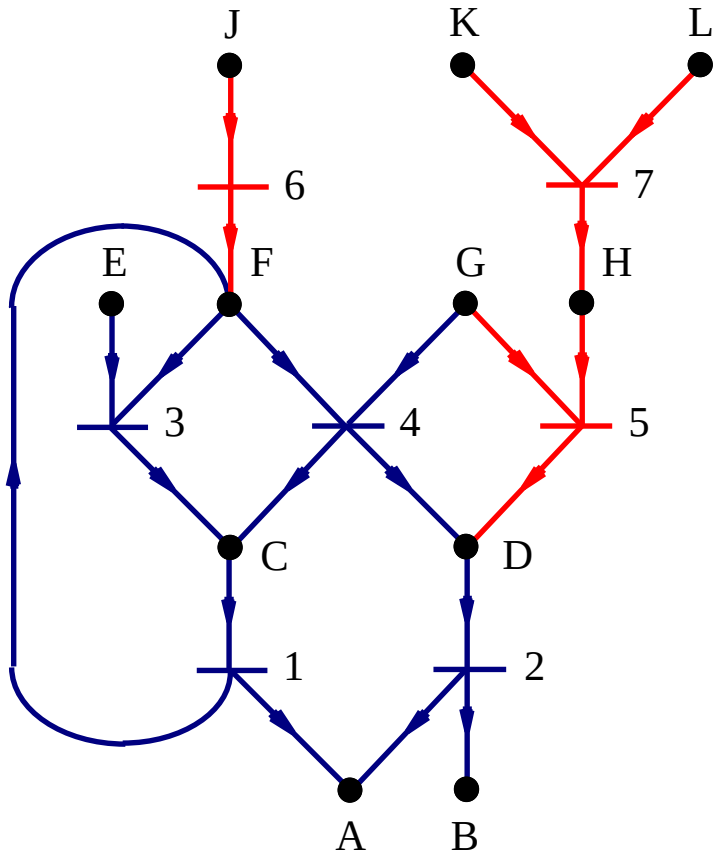
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

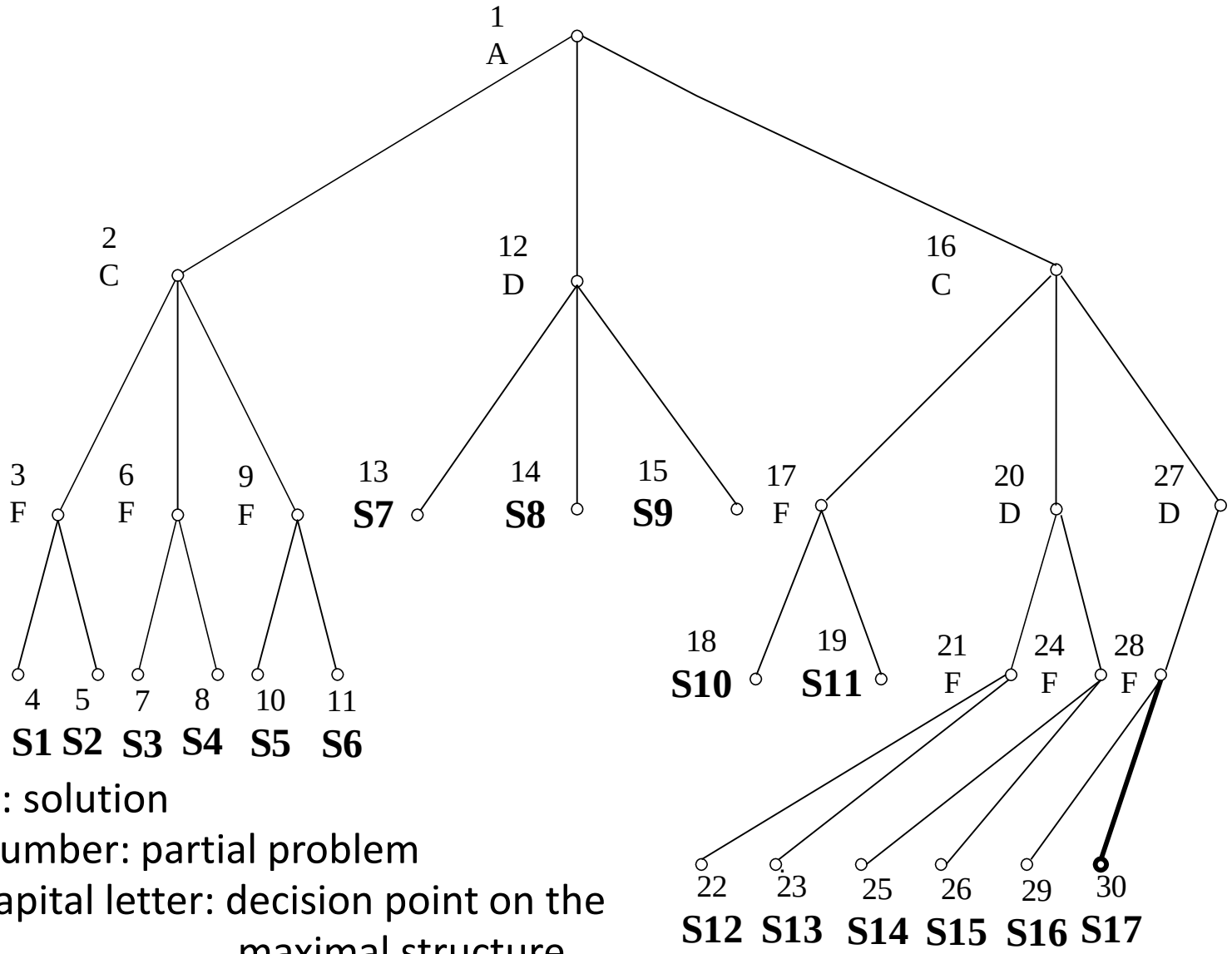
Partial problem #28





Partial problem #29 (S16)

Enumeration (search) tree (worst case)

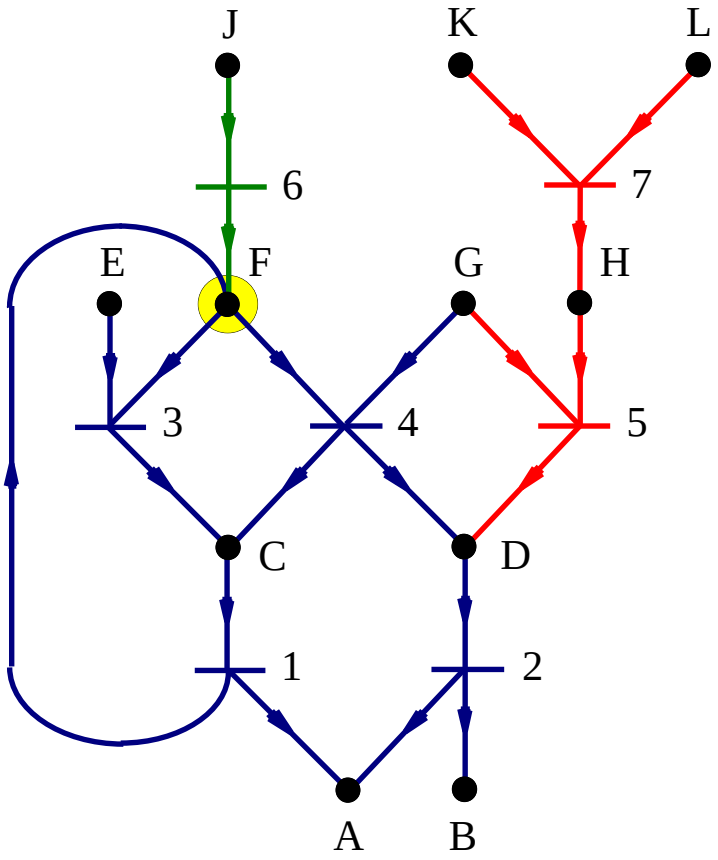


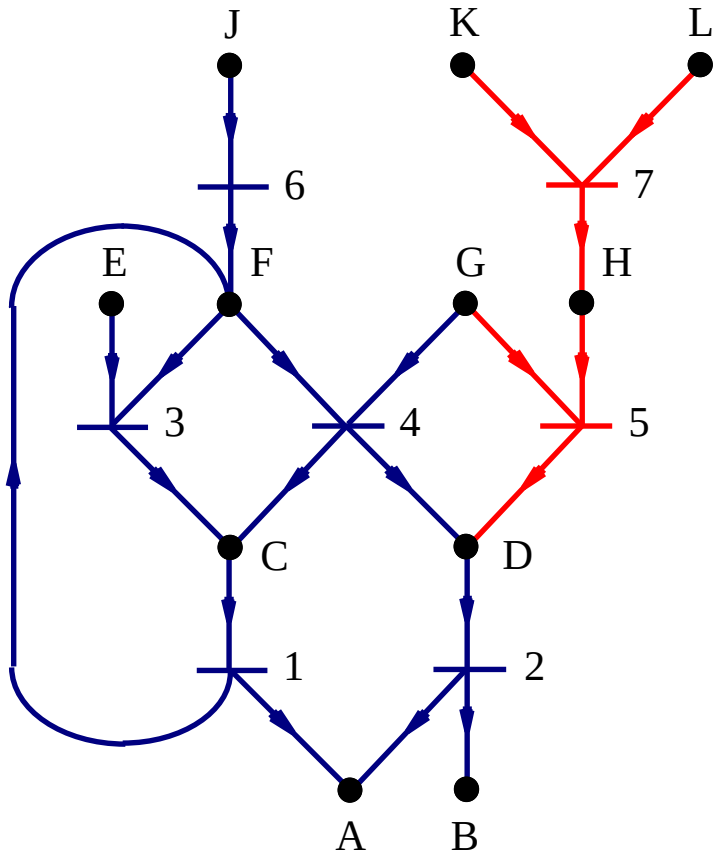
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

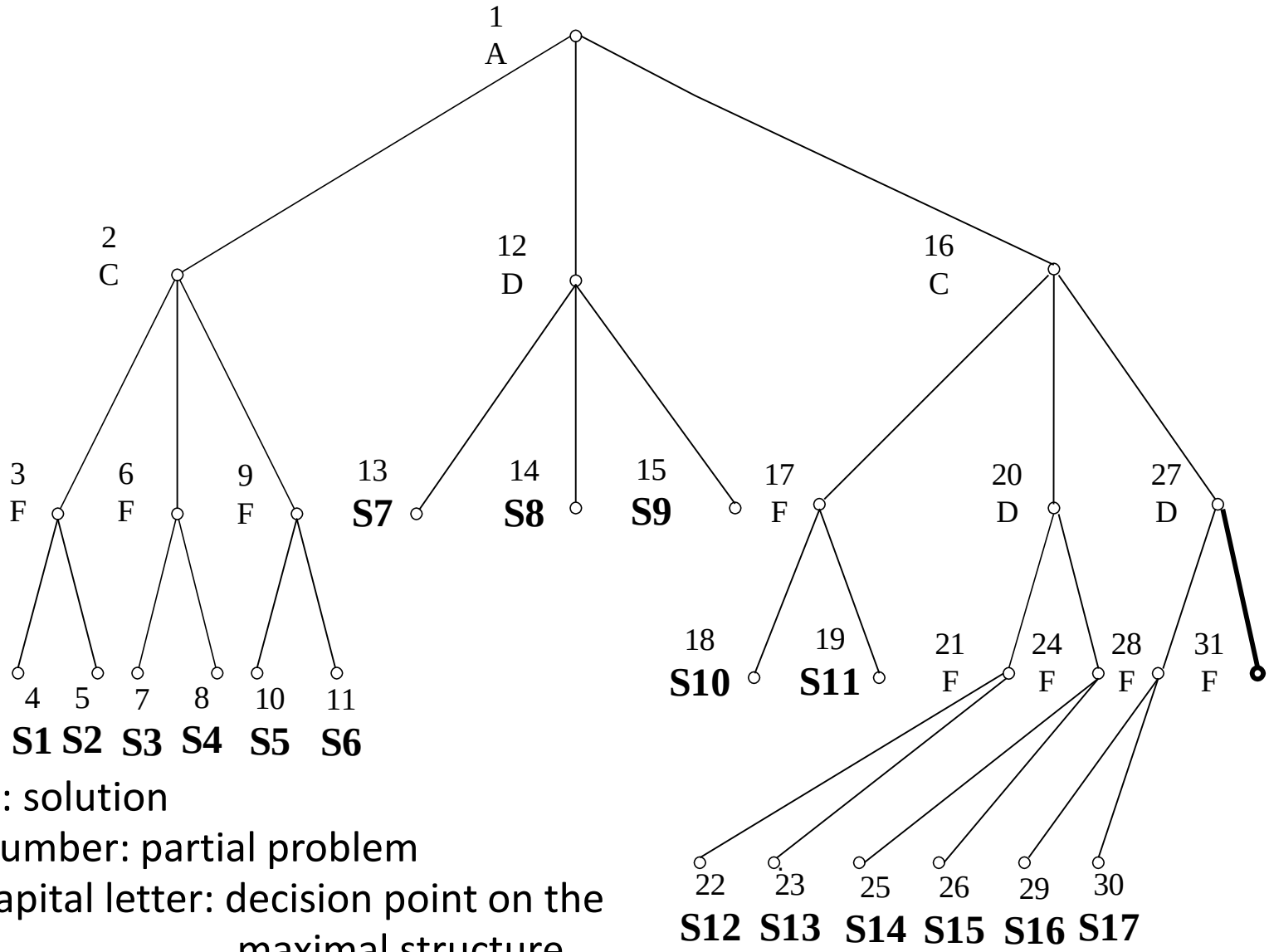
Partial problem #28





Partial problem #30 (S17)

Enumeration (search) tree (worst case)

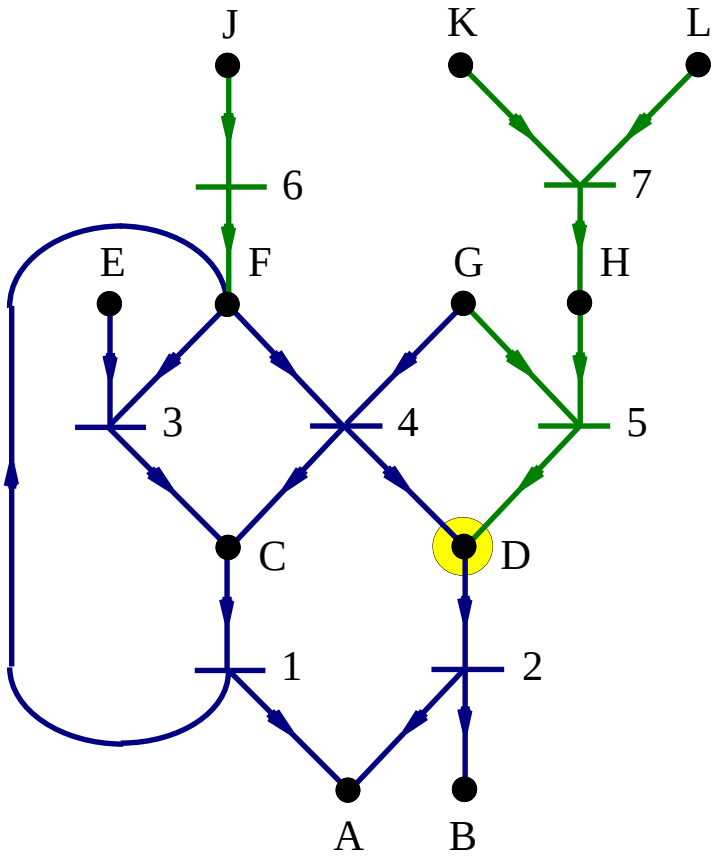


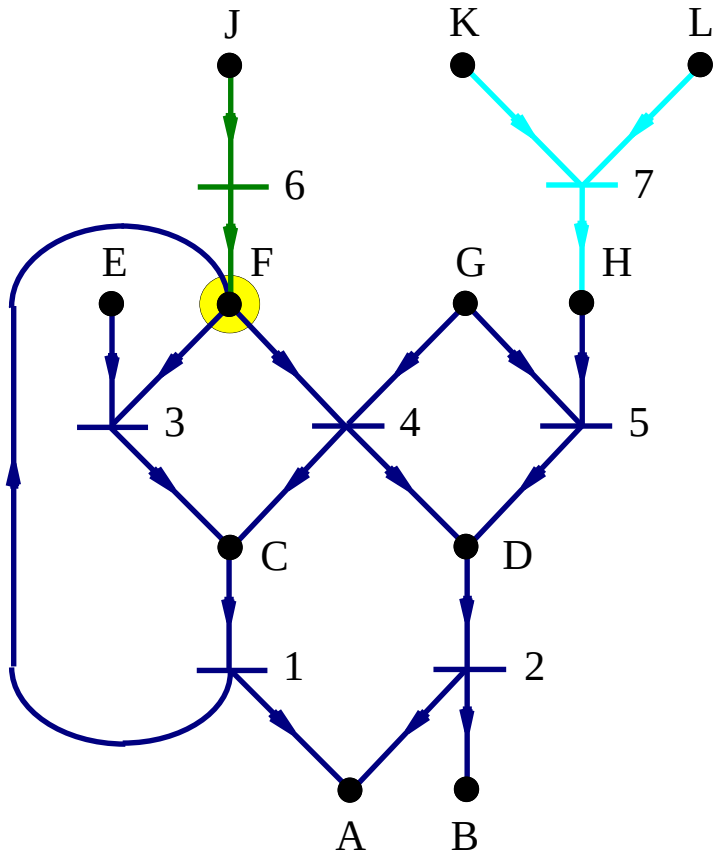
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

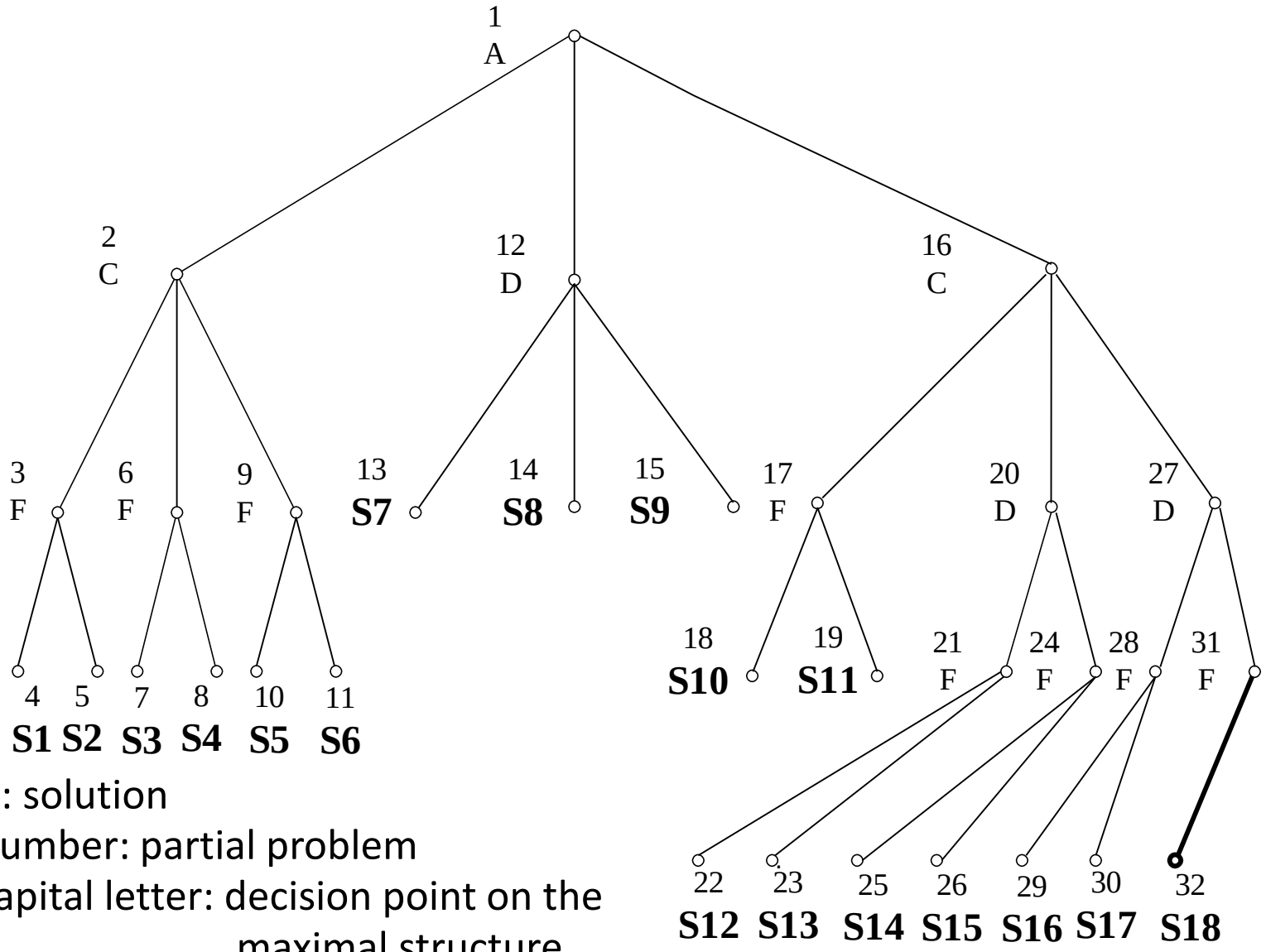
Partial problem #27





Partial problem #31

Enumeration (search) tree (worst case)

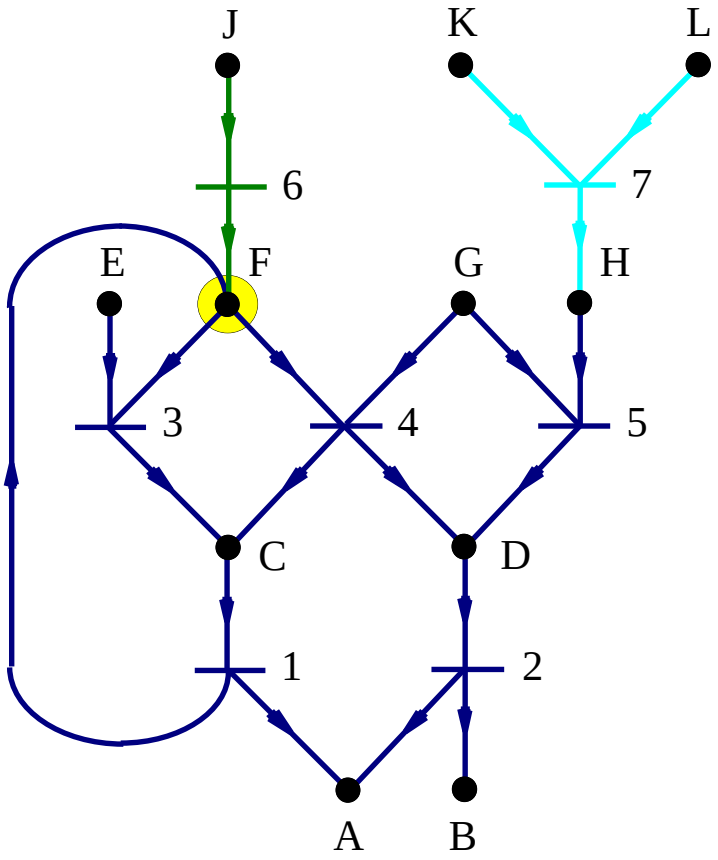


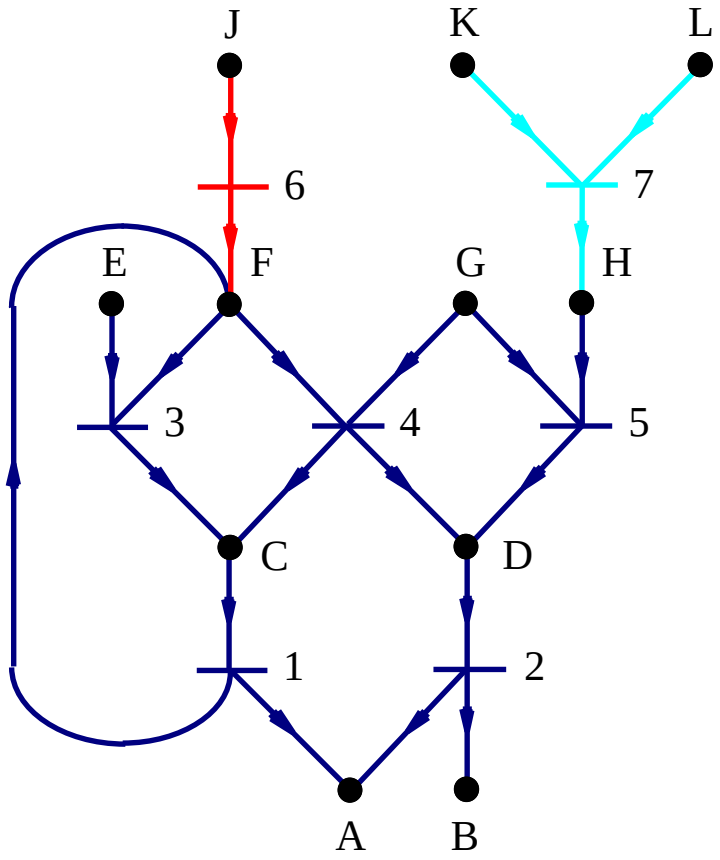
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

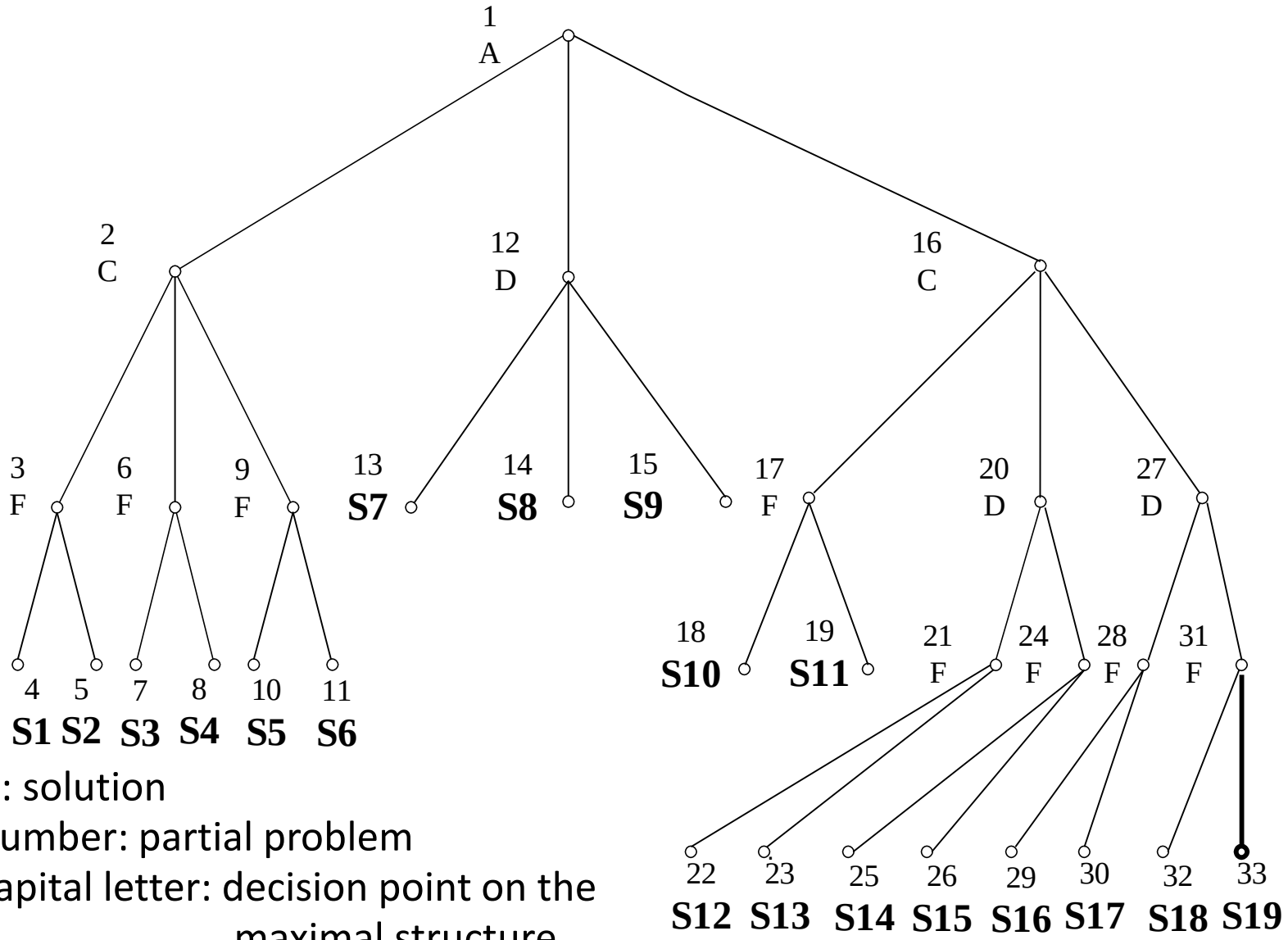
Partial problem #31





Partial problem #32 (S18)

Enumeration (search) tree (worst case)

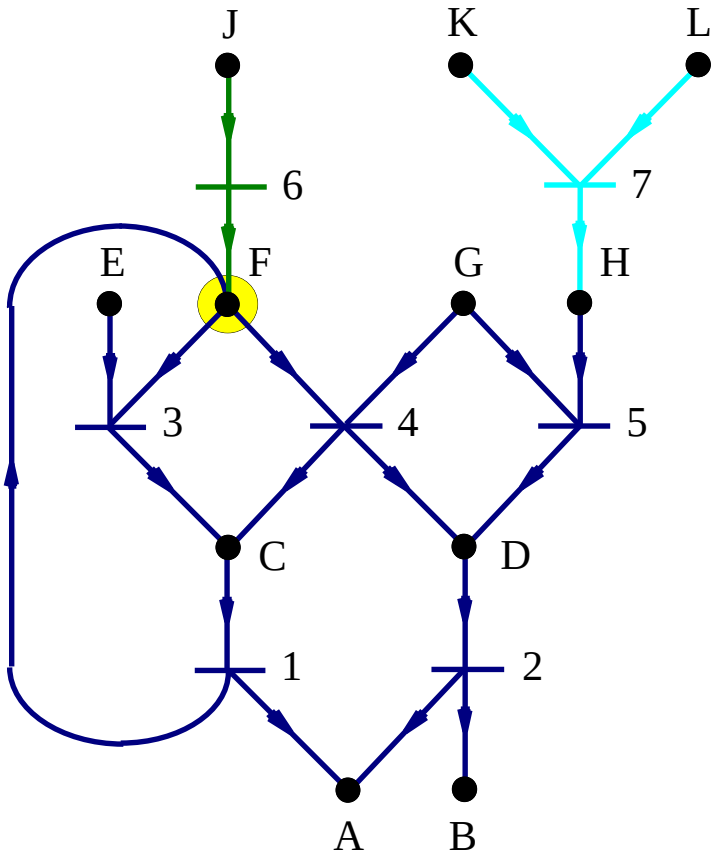


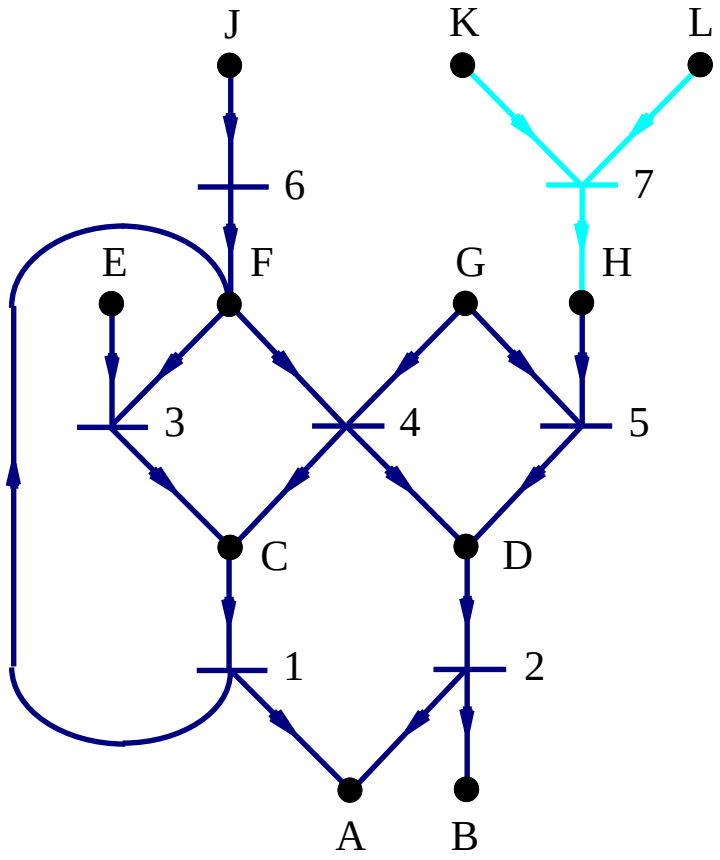
Si: solution

Number: partial problem

Capital letter: decision point on the maximal structure

Partial problem #31

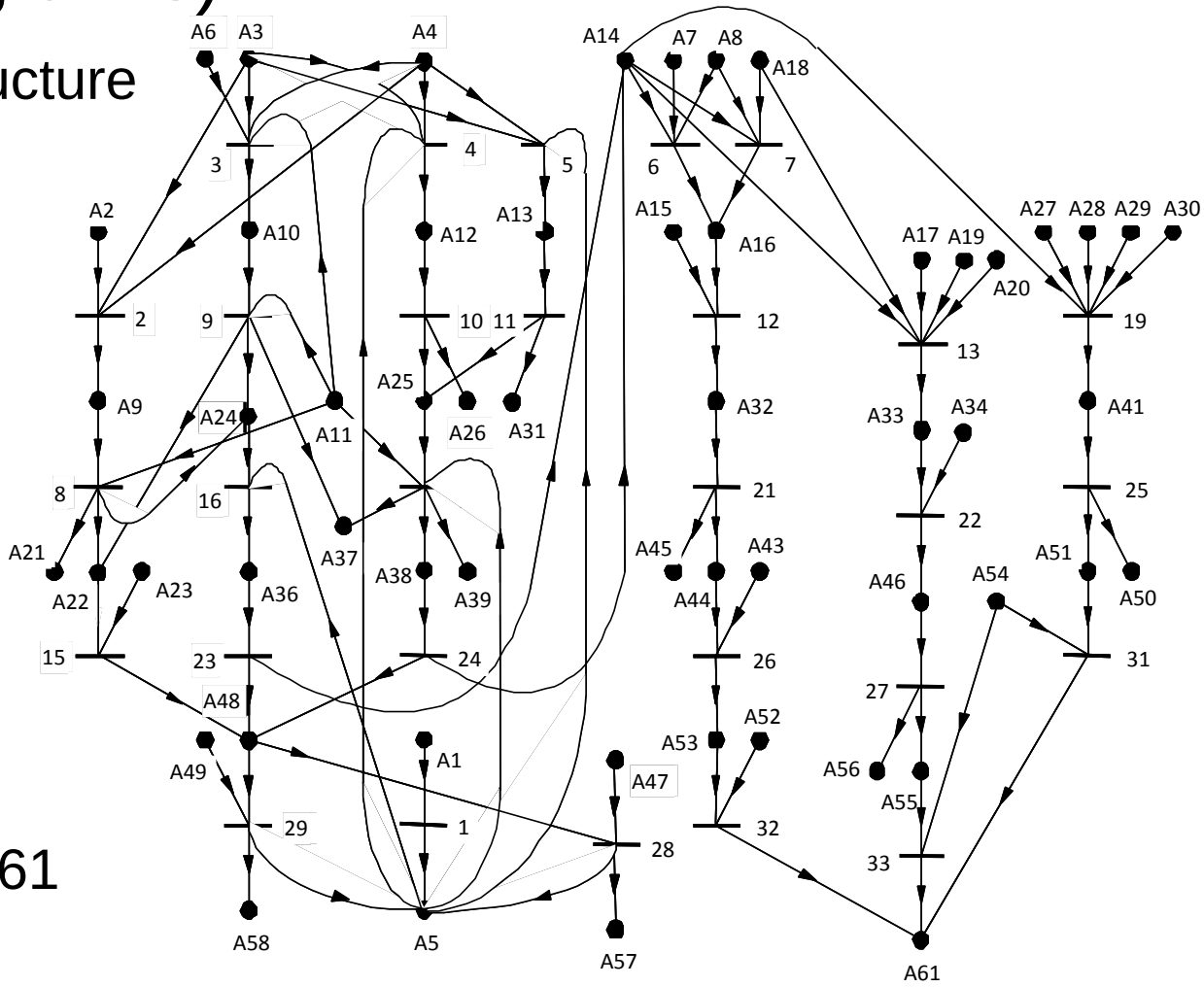




Partial problem #33 (S19)

EXAMPLE PNS 2 (Industrial synthesis problem with 35 operating units)

Maximal structure



Product: A61

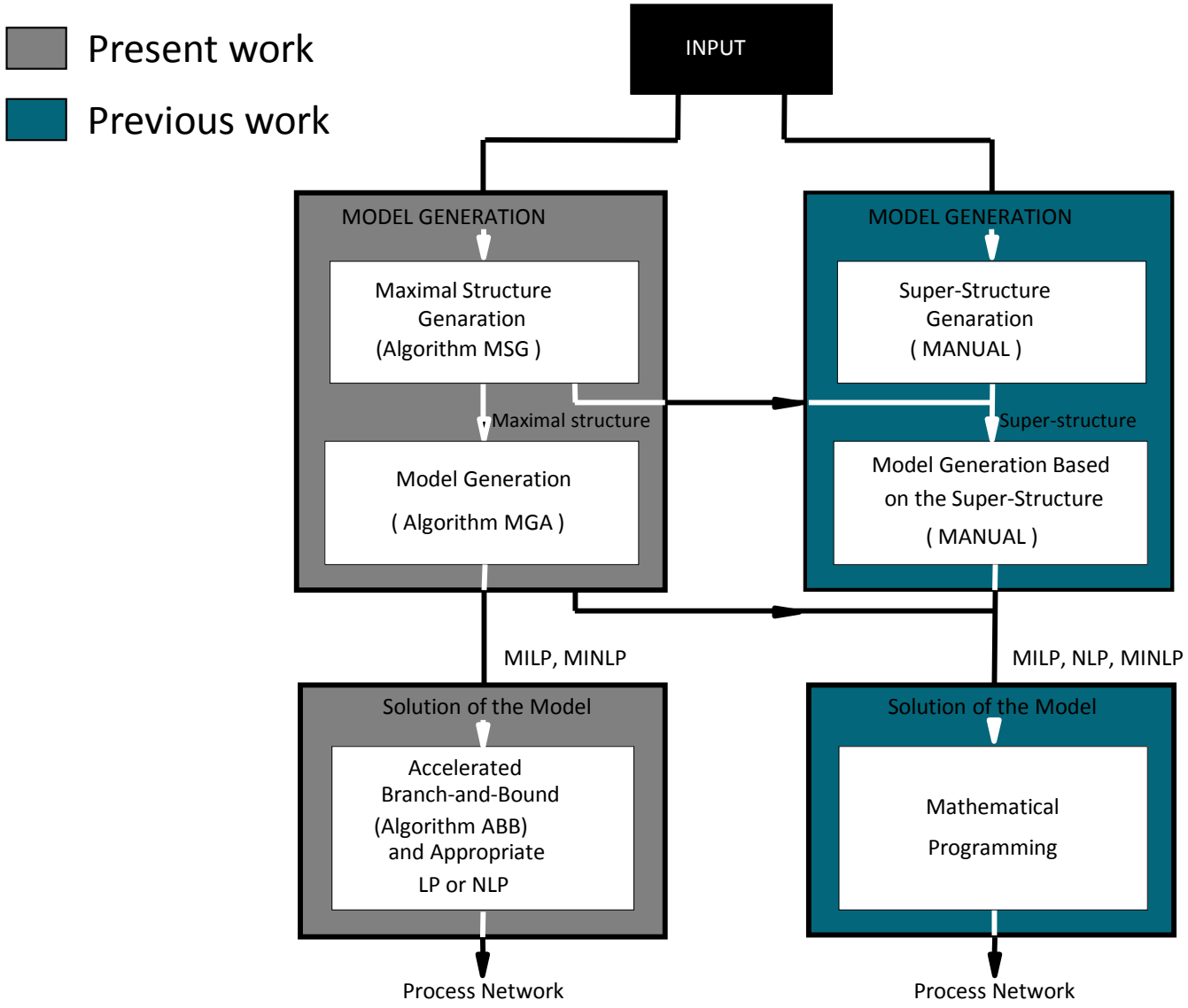
Computational effort required by the basic and accelerated branch-and-bound algorithms in the worst case for Example PNS 2.

Number of partial problems:

Branch-and-bound algorithm: 130 million

Accelerated branch-and-bound algorithm: 8008

PROCESS NETWORK SYNTHESIS



SOLUTION OF AN INDRUSTRIAL RETROFIT SYNTHESIS PROBLEM BY THE ACCELERATED BRANCH-AND-BOUND ALGORITHM: OPTIMAL WATER RECYCLING SYSTEM FOR A NITROCELLULOSE PROCESS

PROBLEM SPECIFICATION

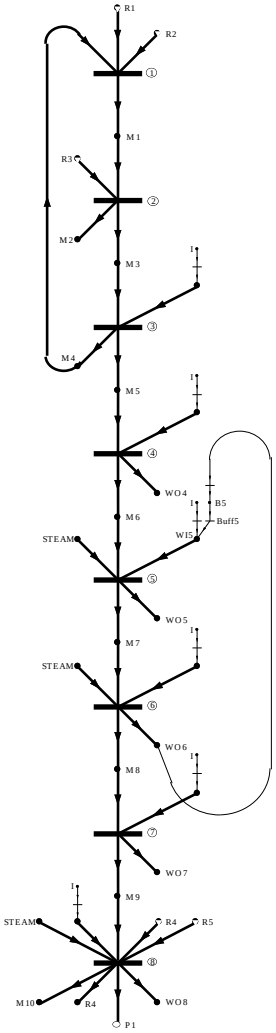
- Streams of water with different quality and quantity are generated in various places of the process.
- The temperatures of these streams are diverse.
- The distances between any pair of operating units vary; this affects the cost of piping.
- The process is semicontinuous; therefore, a buffer has to be installed at an operating unit if water is recycled.

PROBLEM SPECIFICATION

(Cont'd)

- An operating unit may accept only a given subset of the available streams of water with different quality.
- Industrial water can be used at any operating unit.
- Steam is used to heat recycled water if necessary.
- Retrofitting.
- The objective function includes the cost of industrial water and energy (steam), operating cost, and the investment cost of retrofitting (e.g., new piping).

Original nitrocellulose process



Notation

operating unit

material or steam

raw material

product

General features of the existing nitrocellulose process

- Semicontinuous
- Operating units:

  mixing (tank)
  reaction (reactor)
  separation (centrifuge)
  washing (autoclave)
  steaming (tank)
  high pressure steaming (autoclave)
  washing (tank)
  forming (autoclave)
- Water consumption: 166.5 m^3/t

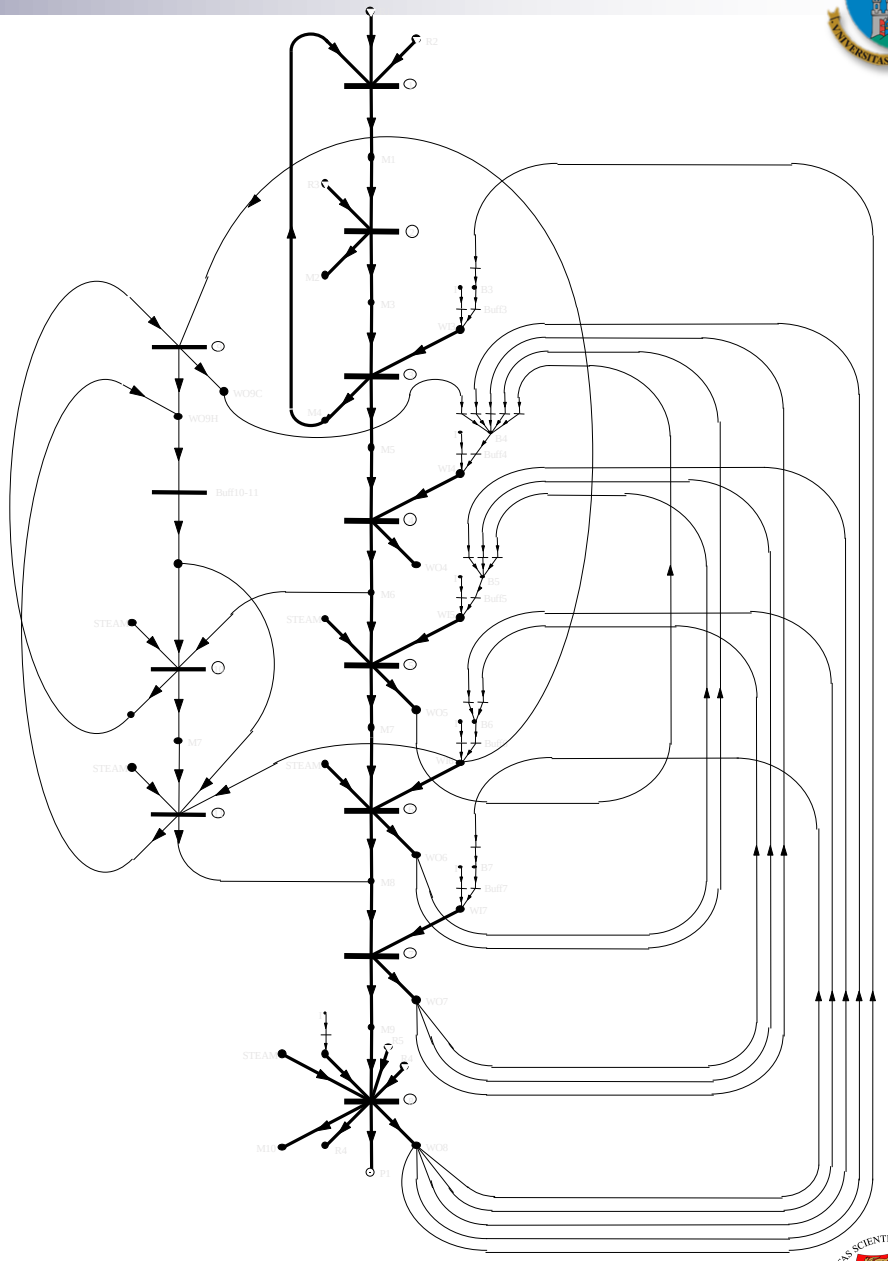
Possible improvements

additional water recycling

modified process structure

STEP 1.
Maximal Structure Generation.

Maximal structure



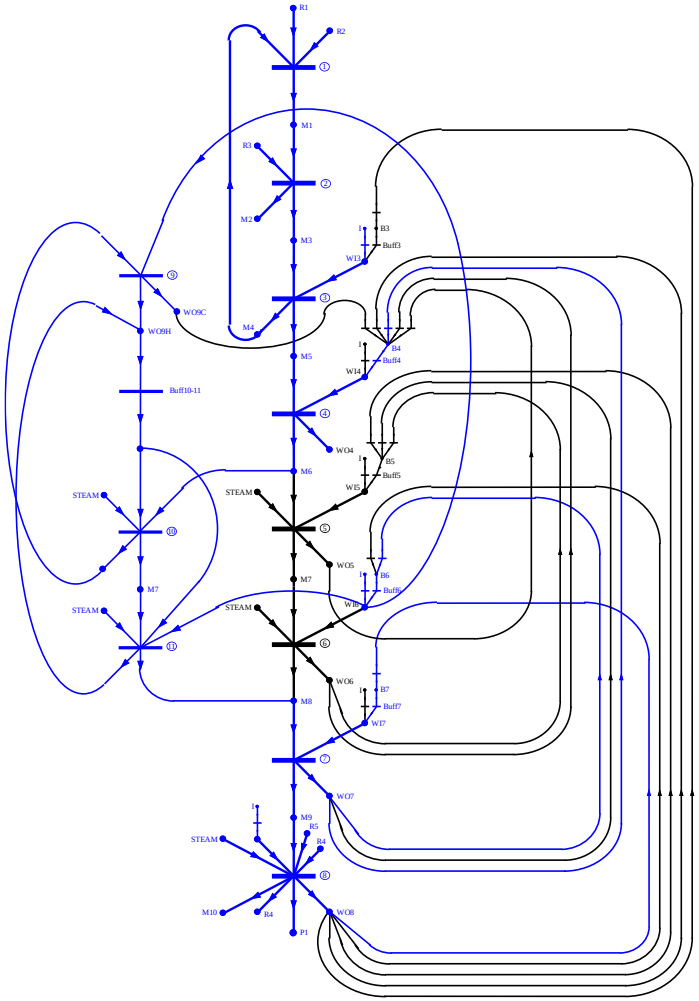
Notation

- operating unit
- material or steam
- raw material
- product

STEP 2.

Generation of the optimal or n-best solutions by the accelerated branch-and-bound algorithm.

Optimal structure



Notation

- operating unit
- material or steam
- raw material
- product
- selected for optimal
- ignored for optimal

COMPARISON OF DIFFERENT SOLUTIONS

	Energy cost saving	Water cost saving	Investment cost	Total cost
Optimal	2592	1440	1130	6488
Second best	2592	1566	1260	6492
Third best	2592	1420	1130	6508
Fourth best	2592	1546	1260	6512
Fifth best	2592	1440	1195	6553
Recent				9390

INTEGRATED SYNTHESIS OF PROCESS AND HEAT EXCHANGER NETWORKS

INTRODUCTION

- Process synthesis – process integration
  - The purpose of process integration is to combine available or planned systems for better performance, e.g., for energy conservation, or pollution reduction, or cost reduction.
  - Process integration usually affects the networks or structures of the systems.
  - For process integration, a convenient process synthesis method is required.

- Sources of difficulties
  - The combination of already complex problems, i.e., the integration of complex design (synthesis) subproblems.
  - Process integration frequently involves at least two classes of synthesis problems.
- The available synthesis methods focus on certain classes of problems, e.g., the synthesis of
  - separation networks.
  - heat exchanger networks.
  - reactor networks.

WHY AVAILABLE SYNTHESIS METHODS CANNOT BE COMBINED TO PERFORM INTEGRATED PROCESS SYNTHESIS

Illustration:

Design of processing systems with heat integration

- The processing system is to be designed as a PNS problem.
- The heat-exchanger network is to be designed as a HEN synthesis (HENS) problem.
- PNS and HENS must be integrated into a super synthesis (to reach the global optimum).
- The available HENS methods assume that the hot and cold streams are specified a priori; it is unsuitable for PNS.

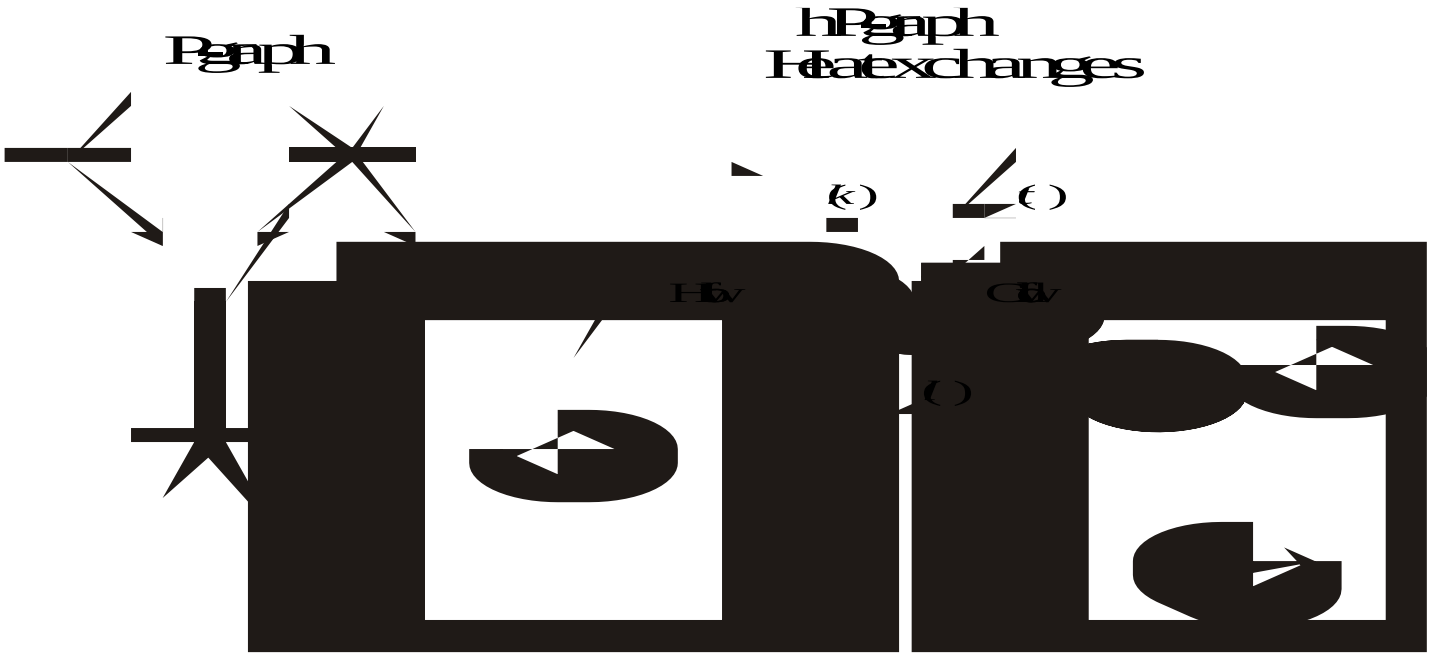
PROPOSED METHOD FOR THE INTEGRATION OF PNS AND HENS

Outline of the Method

- New structure representation as an extension of P-graph.
- A highly effective combinatorial method (algorithm ABB) controls the procedure.
- The mathematical model of the HENS problem is integrated into the mathematical model of a partial problem of PNS generated by algorithm ABB.

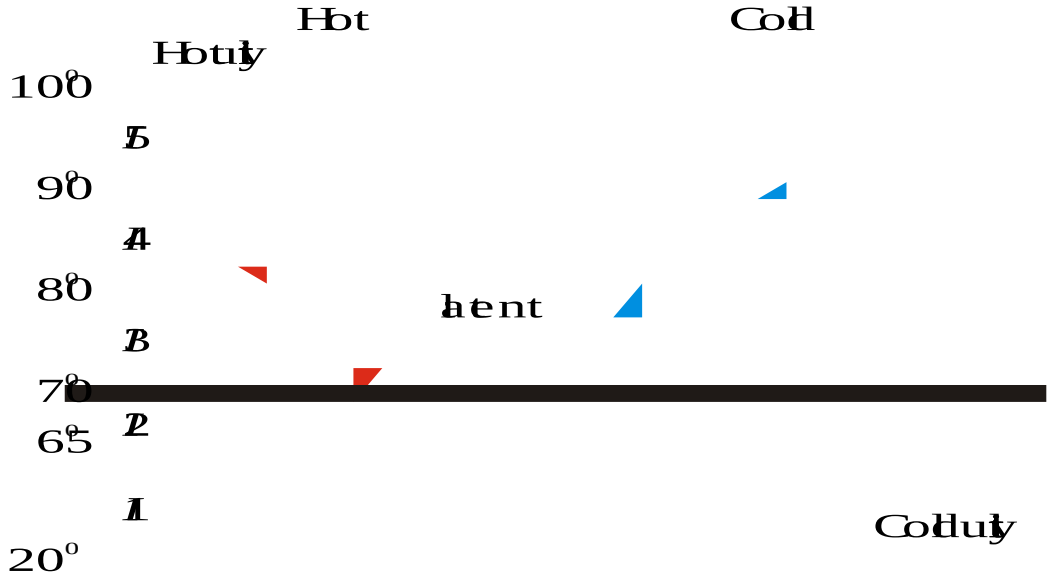
STRUCTURE REPRESENTATION

Heat streams

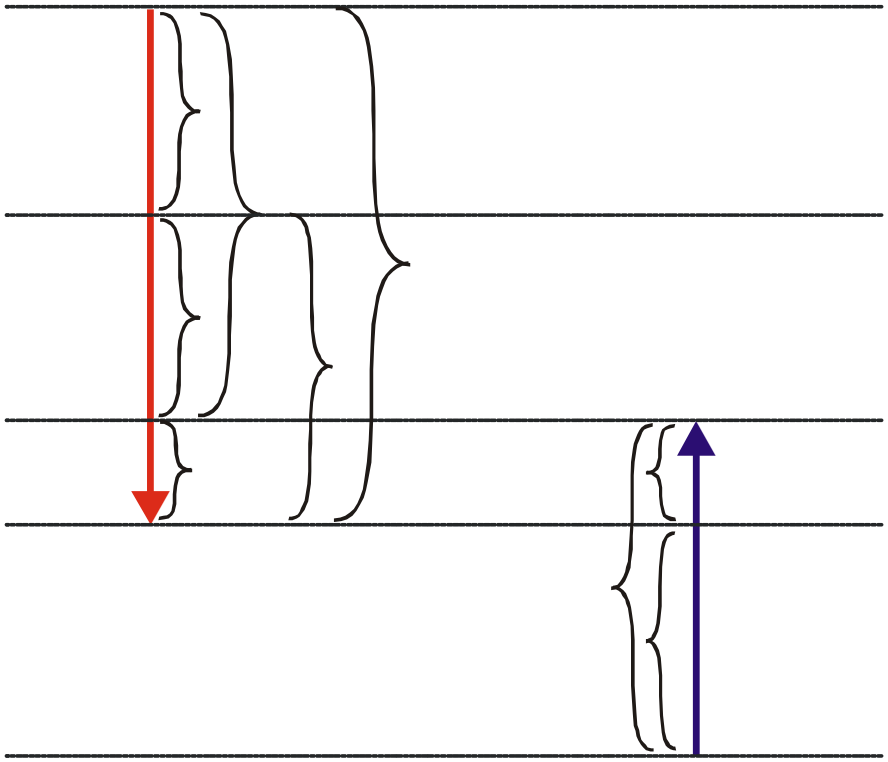


Latent heat

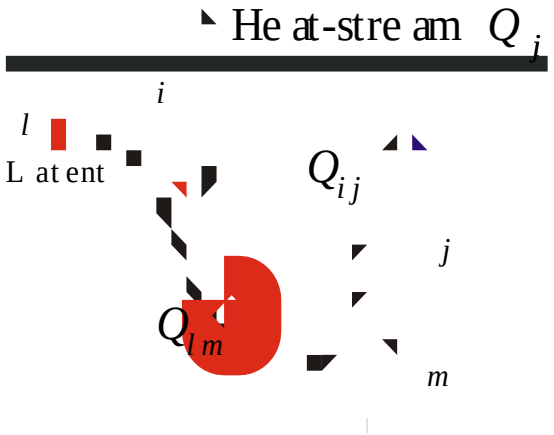
Temperature intervals



Temperature intervals for potential connections



Heat exchangers defined by matching intervals



FORMAL DESCRIPTION

Specific operating cost of heat transfer:

$$c_{ij}(Q_{ij}) = A_{ij} \frac{1}{U_{ij} LMTD_{ij}} Q_{ij}$$

where

Q_{ij} : heat transferred between streams i and streams j.

A_{ij} : unit cost of heat exchanger area between streams i and j.

U_{ij} : heat transfer coefficient between streams i and streams j.

Mathematical model for process synthesis including heat integration

$$\min\left[\sum_{o_j \in O} f_j(y_j, x_j) + \sum_{\substack{i \in Hot,\, j \in Cold \\ I(i) > I(j)}} c_{ij}(Q_{ij}) + \sum_{i \in Hot} \sum_{j \in Cold} CU_j Q_{ij} + \sum_{j \in Cold} \sum_{i \in Hot} HU_i Q_{ij}\right]$$

Constraints on the structure

$$g_k(y_k, o_k, z_k) \le 0, \qquad o_k \in O$$

$$g_i'(m_i) \le 0, \qquad m_i \in M$$

Constraints on the heat balance

$$QB(i) = 0, \qquad i \in Hot \cup Cold$$

where

O: set of operating units

y_k: vector of binary variables for expressing existence (1) or absence (0) of the operating units k

z_j: size of operating unit o_j

CU_j: the unit-cost of the j-th cold utility

HU_i: the unit-cost of the i-th hot utility

EXAMPLE

PNS Part of the Problem Definition

- Product

Name	Req. flow [t/year]
*M*1	100.0

- Raw materials

Name	Price[USD/t]	Max. flow [t/year]
*M*5	140	Unlimited
*M*7	900	Unlimited
*M*9	650	Unlimited
*M*10	500	Unlimited
*M*11	700	Unlimited

Operating units

The linear mathematical models of the operating units: the ratio of the flow rates of the input and output streams of an operating unit is fixed (the relative flow rate is in brackets in the following table).

#	Input streams	Output streams
1	*M*3(3)	*M*1(2), *M*6(1)
2	*M*4(1.5)	*M*1(1), *M*2(0.5)
3	*M*5(1), *M*6(1)	*M*3(2)
4	*M*6(0.3), *M*7(1.7)	*M*3(1), *M*4(1)
5	*M*7(2), *M*8(1)	*M*4(3)
6	*M*9(1)	*M*6(1)
7	*M*10(1.2), *M*11(0.8)	*M*8(2)

Cost Parameters of the Operating Units (MILP Model)

The cost function of an operating unit:

$C_i = A_i + B_i X_i$

where Xi is the relative "size" of operating unit *i*.

Unit	Investment Cost	Operating Cost
1	7,500	20
2	6,000	200
3	10,000	10
4	15,000	10
5	10,000	120
6	3,000	20
7	5,000	160

HENS part of the Problem Definition

- Operating units

#	Latent Heat		Input streams	Output streams
	^{0}C	Param.		
1	-	-	*M*3(3,70)	*M*1(2), *M*6(1,90)
2	-	-	*M*4(1.5)	*M*1(1), *M*2(0.5)
3	80	20	*M*5(1), *M*6(1,80)	*M*3(2,60)
4	-	-	*M*6(0.3), *M*7(1.7)	*M*3(1,90), *M*4(1)
5	-	-	*M*7(2), *M*8(1)	*M*4(3)
6	-	-	*M*9(1)	*M*6(1,55)
7	-	-	*M*10(1.2), *M*11(0.8)	*M*8(2)

Note: The second number in the brackets specifies the temperature of the corresponding stream (if available).

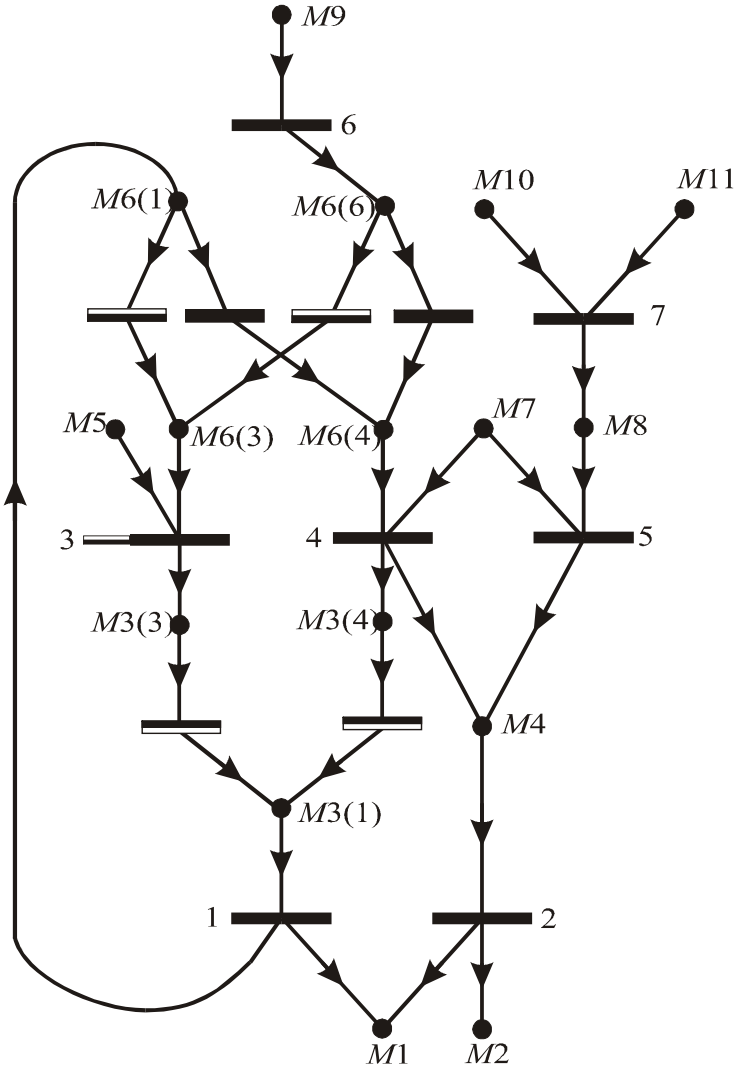
Cost parameters

Cost of heat-exchanger area: 5.0

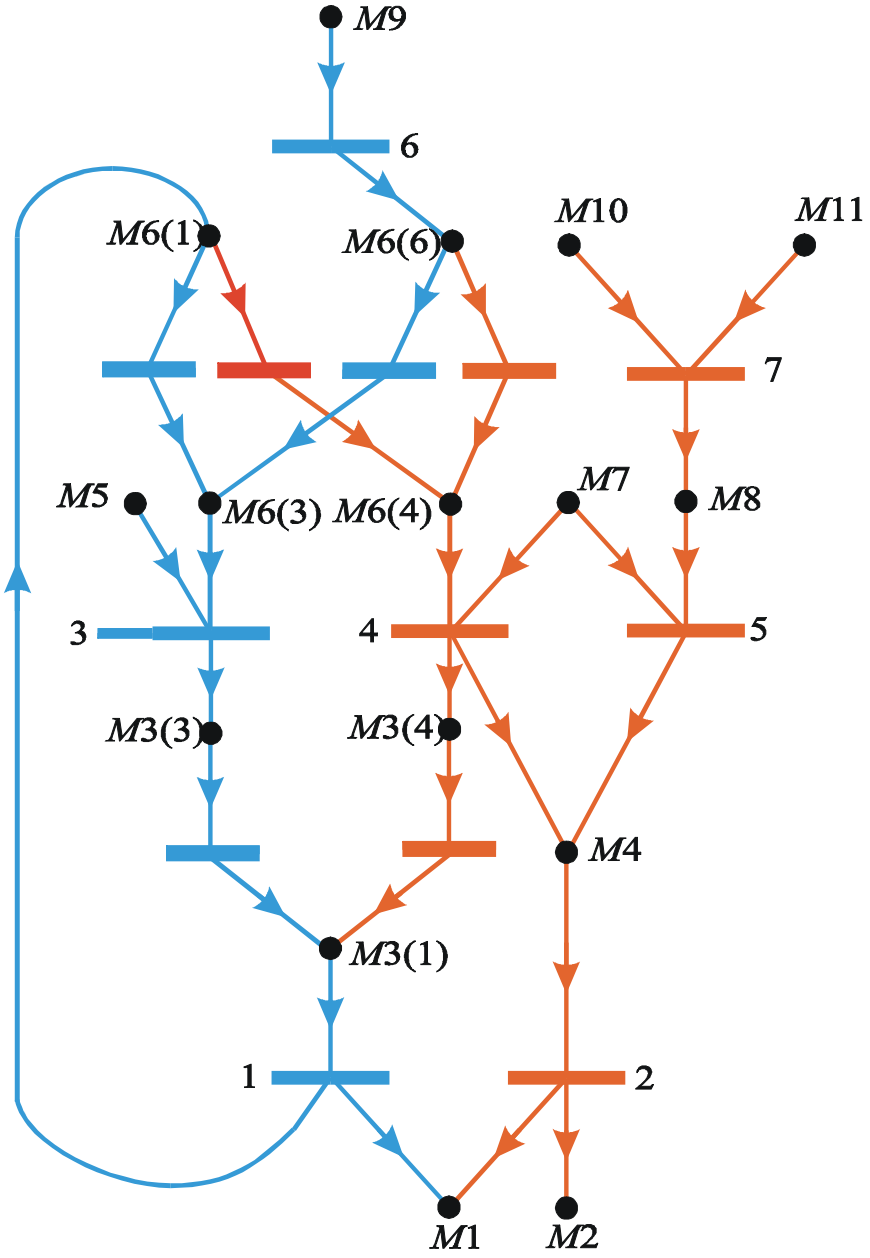
Cost of utility

Utility	Type	Temp.(^{0}C)	Cost
1	Hot	10.0	20.0
2	Cold	100.0	30.0

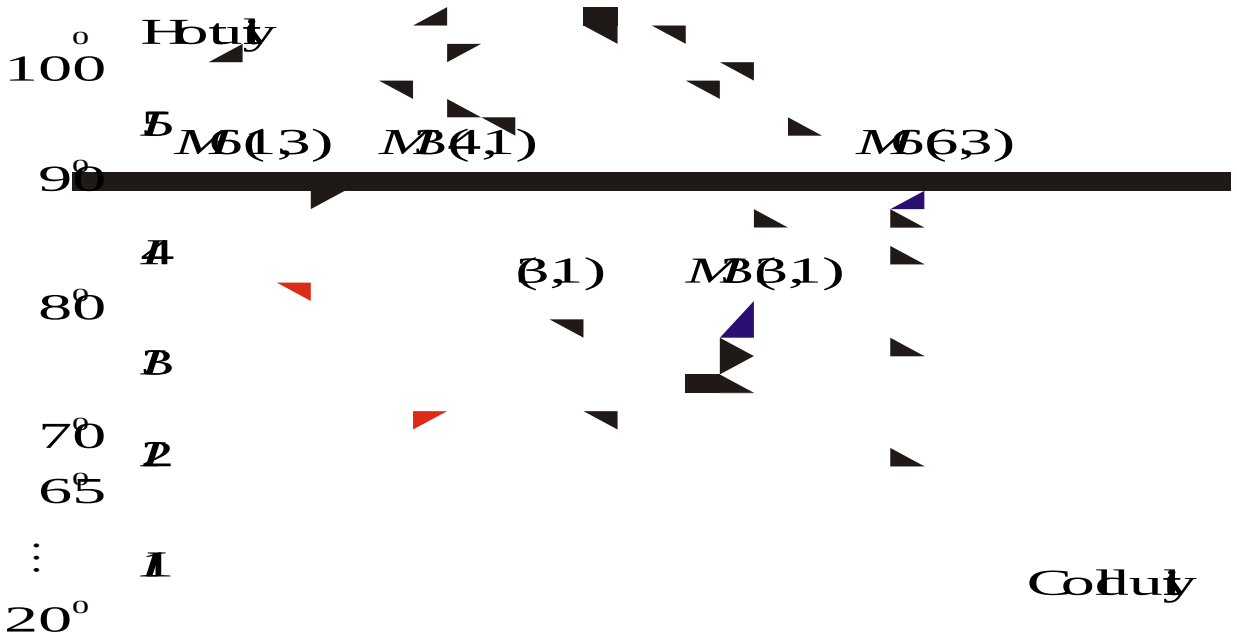
Integrated maximal structure



Optimal structure

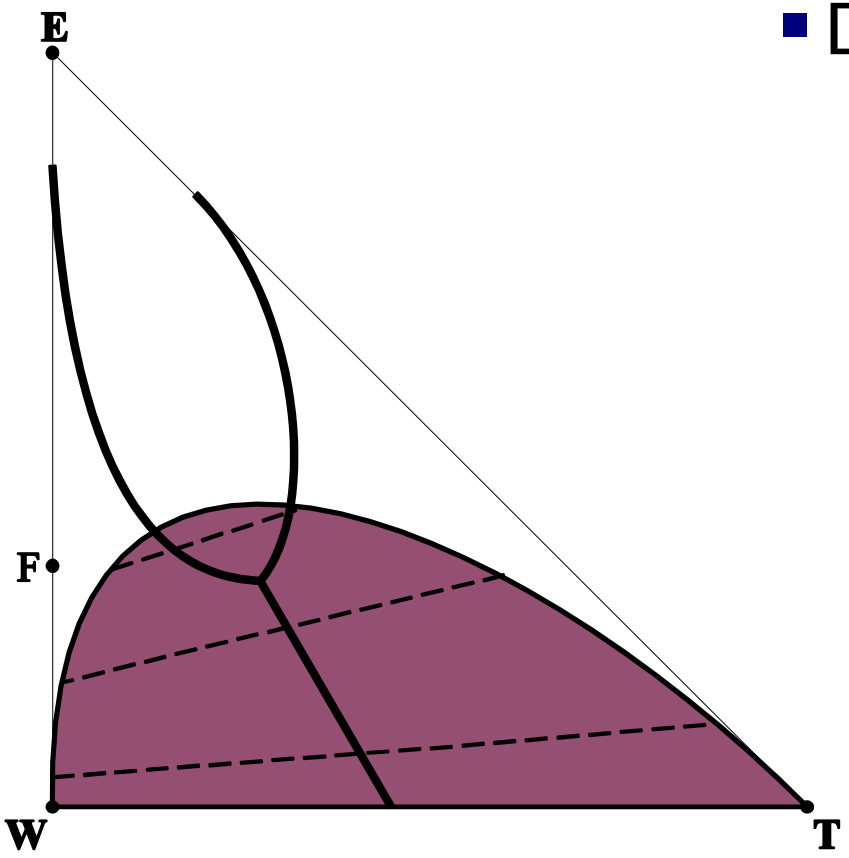


Heat transfer on the optimal structure



ALGORITHMIC SYNTHESIS OF AZEOTROPIC-DISTILLATION SYSTEMS

AZEOTROPIC-DISTILLATION PROBLEM

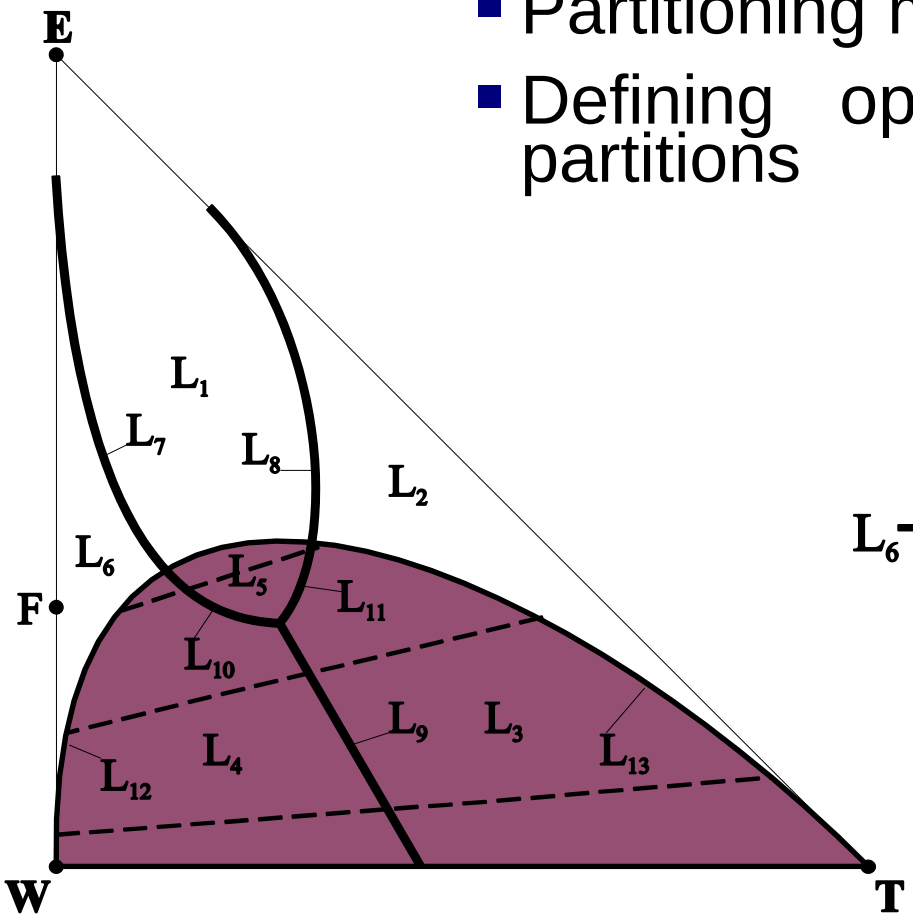


Azeotropic-distillation problem represented by RCM

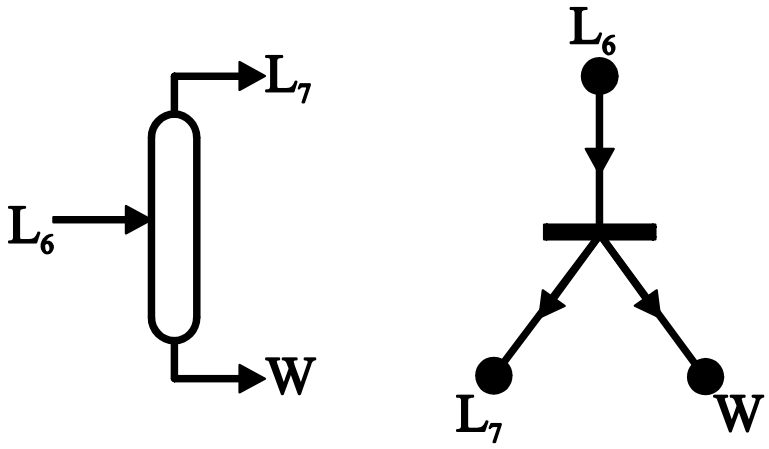
- Defined by
  - Feed (F)
  - Product (E)
  - Distillation boundaries
  - Phase-splitting regions

IDENTIFYING OPERATING UNITS

- Partitioning materials
- Defining operating units among partitions



Lumped materials (L_1..L_{13})

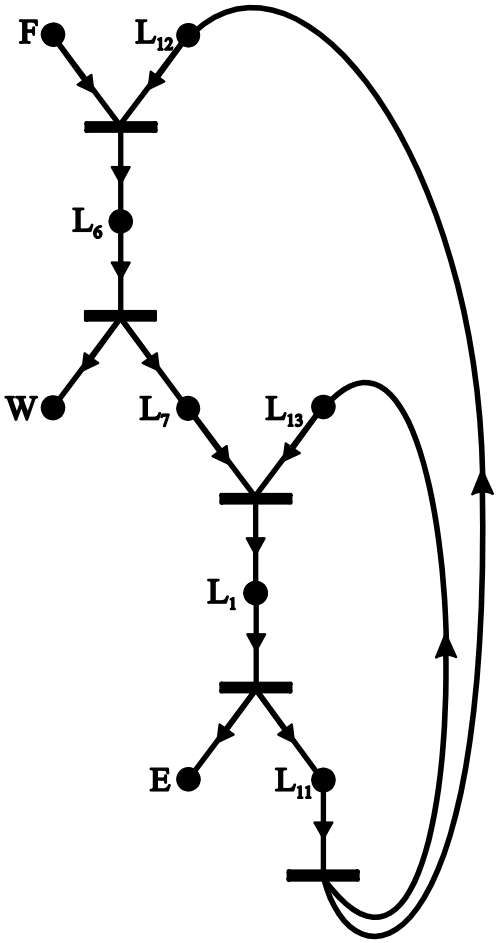


Separator #6 represented by conventional and P-graph.

COMBINATORIAL ALGORITHMS

- Maximal Structure Generator (MSG)
- Solution-structure Generator (SSG)

  (Generates all the combinatorially feasible networks)



Solution-structure #140

REACTION PATHWAY IDENTIFICATION

REACTION PATHWAY IDENTIFICATION PROBLEM

- Given by the stoichiometric equations of:
  - Overall reaction
  - Set of elementary reactions

- Example
  - Overall reaction:
  - $C_4H_{10} \rightleftharpoons C_4H_8 + H_2$
  - Set of elementary reactions:

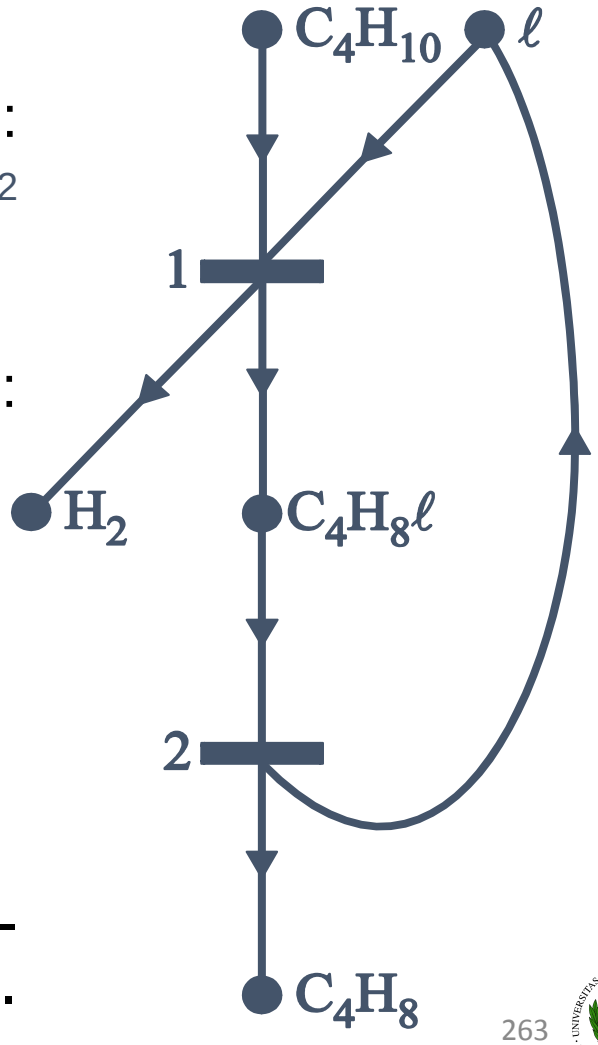
(1) $C_4H_{10} + \ell \rightleftharpoons C_4H_8\ell + H_2$

(2) $C_4H_8\ell \rightleftharpoons C_4H_8 + \ell$

(3) $C_4H_8\ell \rightleftharpoons C_4H_6\ell + H_2$

P-GRAPH REPRESENTATION OF REACTION PATHWAYS

- Example:
  - Elementary reactions:
    (1) $C_4H_{10} + \ell \rightarrow C_4H_8\ell + H_2$
    (2) $C_4H_8\ell \rightarrow C_4H_8 + \ell$
  - Overall reaction:
    $C_4H_{10} \rightarrow C_4H_8 + H_2$



Reaction pathway represented by P-Graph.

SCHEDULING OF MULTIPURPOSE BATCH PLANTS

BUILDING BLOCKS OF THE NEW FRAMEWORK

- New representation technique (S-graph)
- Elementary combinatorial algorithms

REPRESENTATION TECHNIQUE

- Conventional graph representation is convenient for unlimited intermediate storage policy (job-shop).
- It does not represent “non-intermediate storage” policy appropriately.

NEW REPRESENTATION: S-GRAPH

Unified representation for the

- recipe
- intermediate phase of the scheduling procedure
- final schedule

(See Sanmarti *et al*., 1998 for details)

ILLUSTRATION OF S-GRAPH

Initial step (recipe)

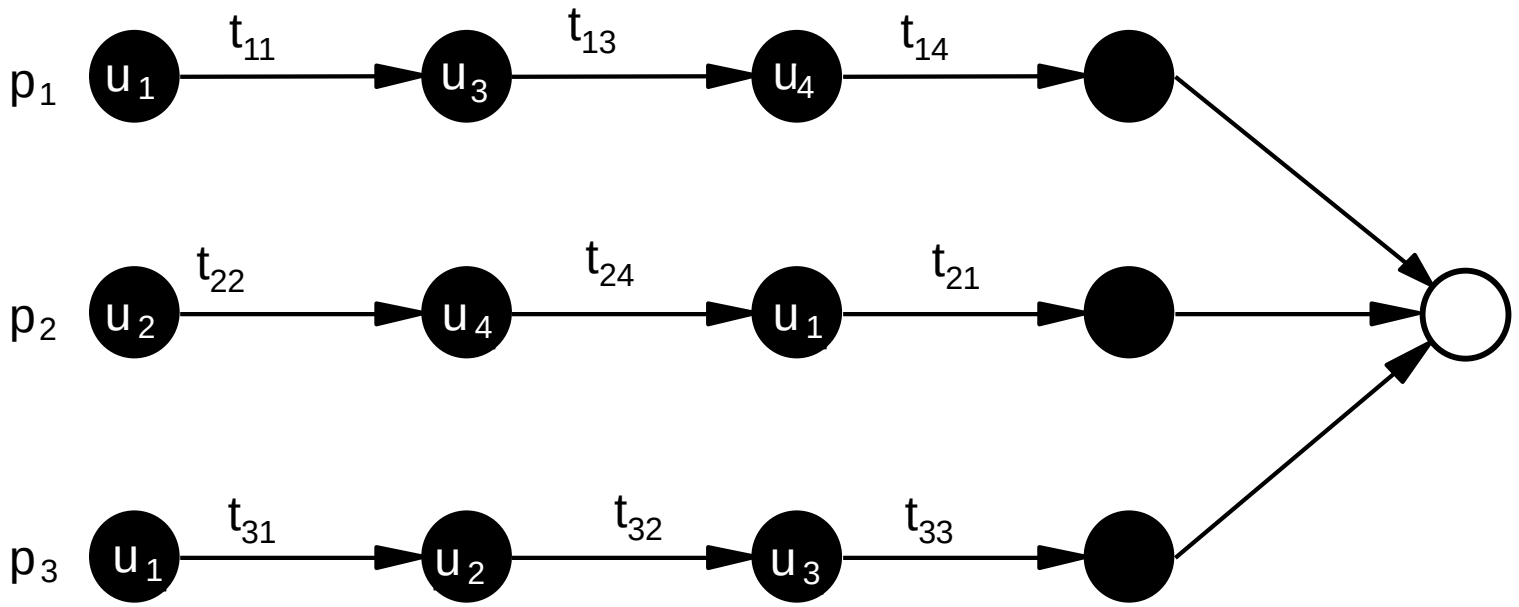


ILLUSTRATION OF S-GRAPH

Step 1.

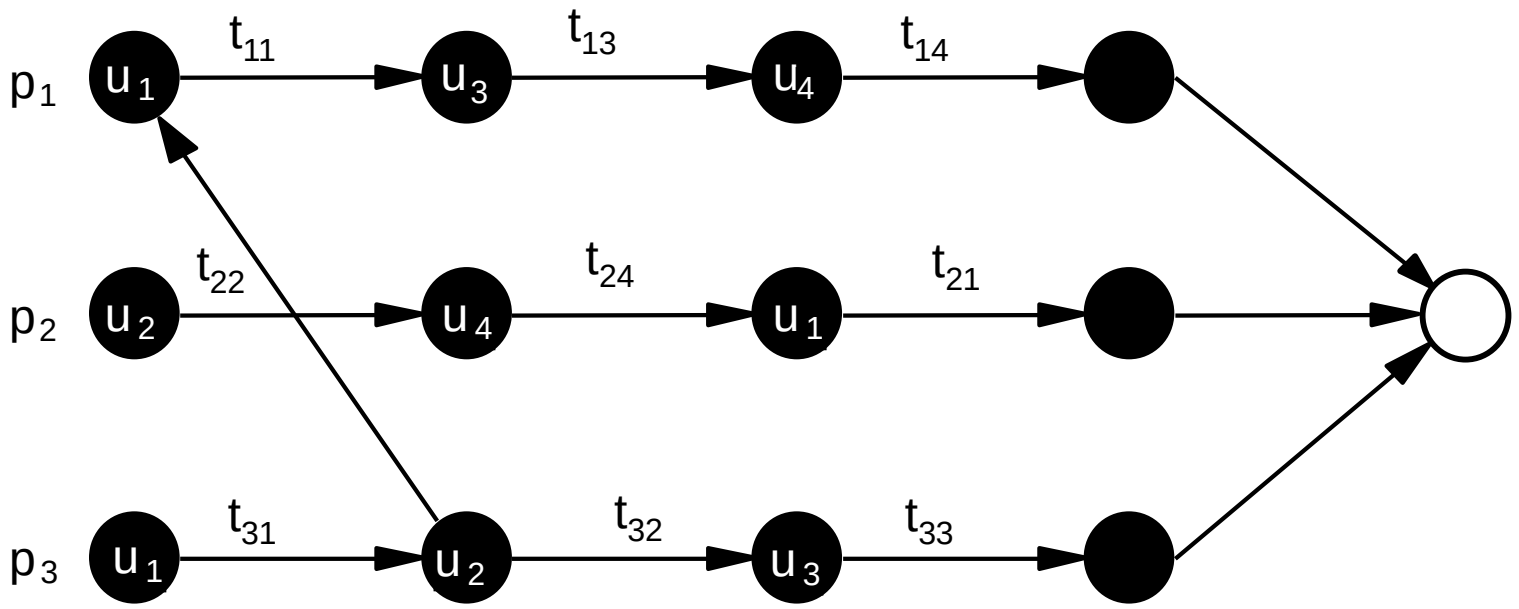


ILLUSTRATION OF S-GRAPH

Step 2.

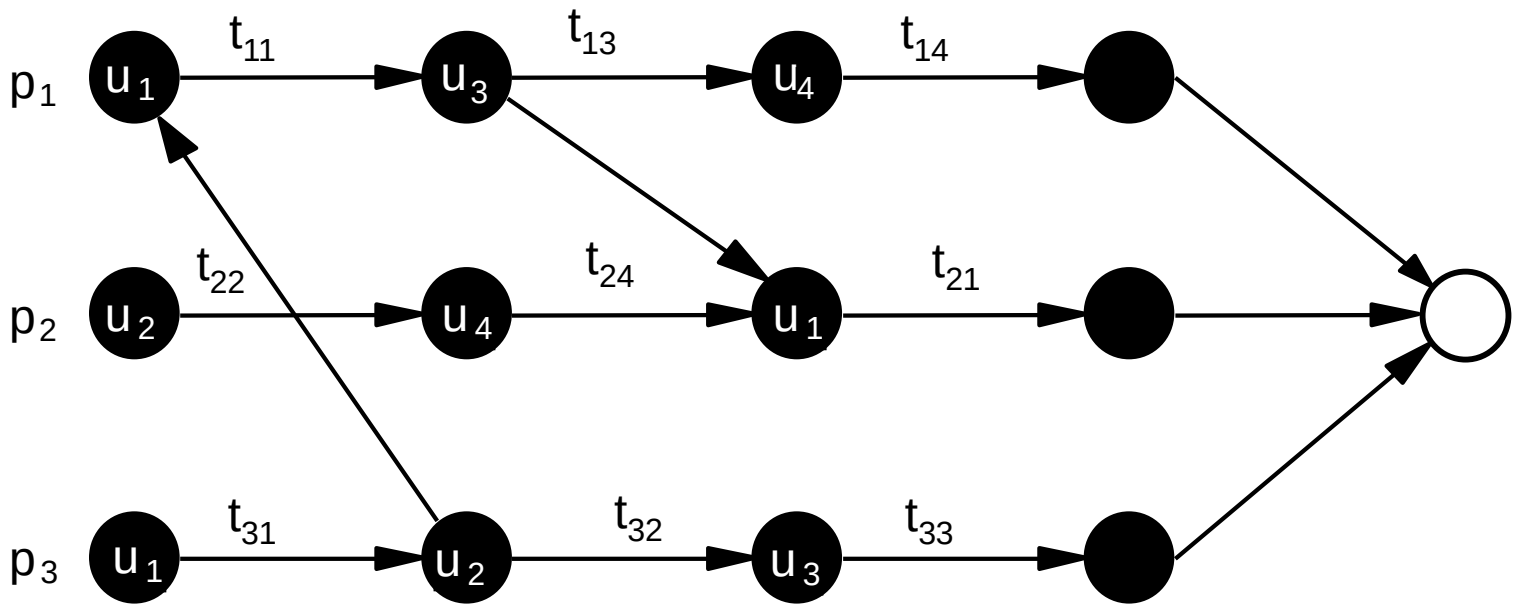
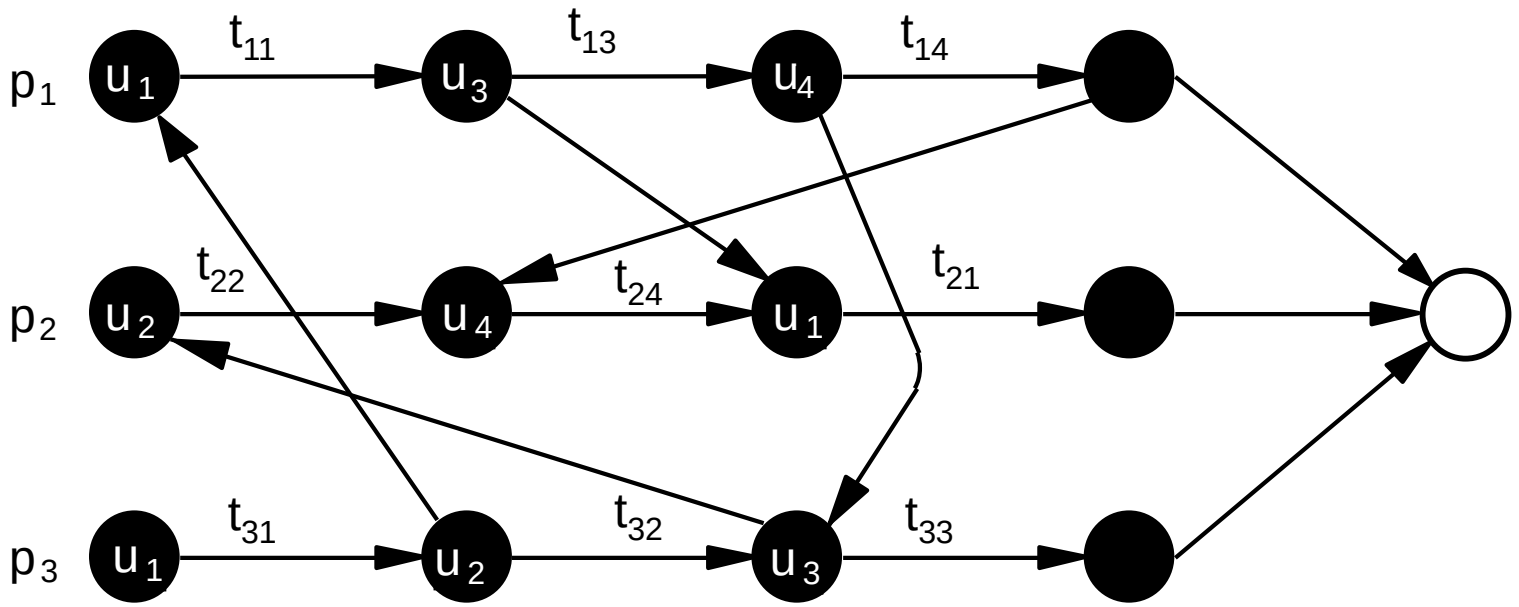


ILLUSTRATION OF S-GRAPH

Final schedule



GENERAL FEATURES OF THE FRAMEWORK

- It serves as base for specific scheduling algorithms
- It takes into account:
  - complex recipe
  - limited waiting time
  - transfer time
  - due date
  - concurrent equipments
- Can be integrated with process synthesis

EXAMPLE SCH 1

Number of Equipments

Mixer: 4

Tank: 11

Packing line: 4

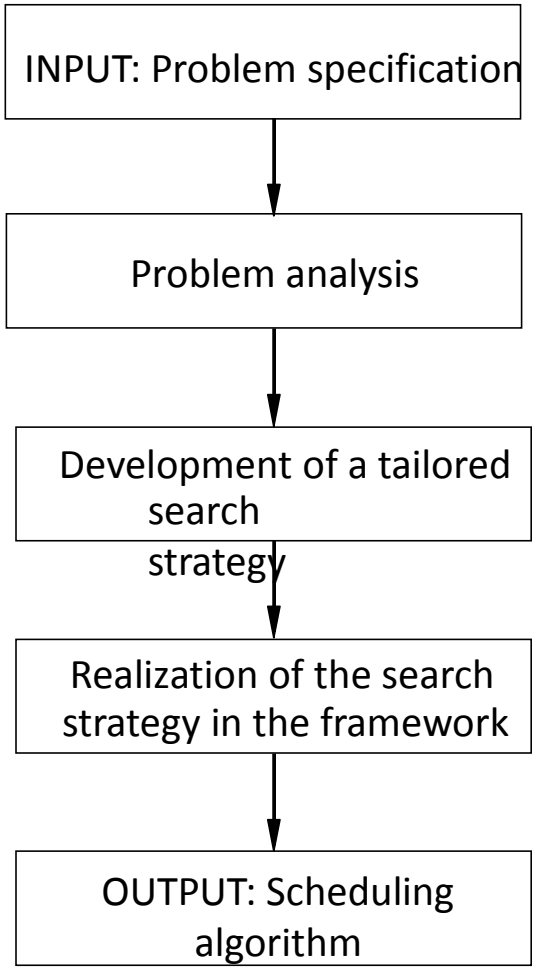
Product	# of batches
P1	2
P2	3
P3	5
P4	2
P5	5
P6	4
P7	5
P8	5
P9	1
P10	1
Total	43

Note: Input is in an Excel file

EXTREMELY COMPLEX SCHEDULING PROBLEMS

- Practical scheduling problems can be difficult to solve because of
  - their size,
  - involvement of continuous and batch operations.
- The new framework serves as a base in the development of effective algorithms for extreme problems.

DEVELOPMENT PROCESS OF A SCHEDULING ALGORITHM



EXAMPLE SCH 2
COMPLEX INDUSTRIAL SCHEDULING PROBLEM

Number of products: 123

Number of Equipments

- Mixer: 5
- Tank: 40
- Packing line: 26

Total number of batches: 334

Note: Running time on PC (333 MHz) is less than 15 minutes.

CONCLUDING REMARK

Combinatorial framework may effectively control the solution procedure of complex process design and operations problems

Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: axioms and theorems. Chem. Eng. Sci. 1992, 47, 1973.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Combinatorial algorithms for process synthesis. Comput. Chem. Eng. 1992, 16, S313.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- http://www.p-graph.com

Synthesis of optimal processes

Introduction

- Algorithm to generate maximal structure (MSG, Maximal Structure Generation)
- Algorithm to generate all combinatorial feasible solutions (SSG, Solution Structure Generation)
- Generate the optimal solution (ABB, Accelerated Branch and Bound)

Notations

- Let $\psi^{-}(\mathrm{o})$ denotes the set of all input materials of operating units included set o

$$\psi^{-}(\mathrm{o}) = \bigcup_{(\alpha,\beta)\in \mathrm{o}} \alpha$$

- Let $\psi^{+}(\mathrm{o})$ denotes the set of all output materials of operating units included set o

$$\psi^{+}(\mathrm{o}) = \bigcup_{(\alpha,\beta)\in \mathrm{o}} \beta$$

- Let $\psi(\mathrm{o})$ denotes the set of all input and output materials of operating units included set o

$$\psi(\mathrm{o}) = \psi^{-}(\mathrm{o}) \cup \psi^{+}(\mathrm{o})$$

Notations

- Let $\varphi^{-}(\mathrm{m})$ denotes the set of operating units producing any material included set m

$$\varphi^{-}(\mathrm{m}) = \{(\alpha, \beta) \in \mathrm{o} : \beta \cap \mathrm{m} \neq \emptyset\}$$

- Let $\varphi^{+}(\mathrm{m})$ denotes the set of operating units consuming any material included set m

$$\varphi^{+}(\mathrm{m}) = \{(\alpha, \beta) \in \mathrm{o} : \alpha \cap \mathrm{m} \neq \emptyset\}$$

- Let $\varphi(\mathrm{m})$ denotes the set of operating units producing or consuming any material included set m

$$\varphi(\mathrm{m}) = \varphi^{-}(\mathrm{m}) \cup \varphi^{+}(\mathrm{m})$$

Notations

- The above functions are valid in case of single material or operating unit
  - For example $\psi^{-}(o_i)$ denotes the set of input materials of operating unit o_i , i.e., $\psi^{-}(o_i) = \psi^{-}(\{o_i\})$

MSG algorithm

Solution structures

- A P-graph can represent the structure of a production system, but an arbitrary S-graph cannot represent the behavior of the system
- To represent a valid $(\mathcal{P}, \mathcal{R}, \mathcal{O})$ synthesis problem, the (m, o) P-graph has to fulfill five combinatorial properties

Axioms

- (S1) Every final product is represented in the graph

$$\mathcal{P} \subseteq \mathrm{m}$$

- (S2) A material type vertex has no input if and only if it represents a raw material

$$\mathrm{m} \setminus \psi^{-}(\mathrm{o}) = \mathrm{m} \cap \mathcal{R}$$

- (S3) Every operating unit type represents an operating unit defined in the synthesis problem

$$\mathrm{o} \in \mathcal{O}$$

- (S4) Every operating unit type vertex has at least one path leading to a vertex representing a final product

$$\forall o_i \in \mathrm{o}, \exists \mathrm{path}[o_i, m_i], \text{ where } m_j \in \mathcal{P}$$

- (S5) If a material type vertex belongs to the graph, it must be an input or an output of at least one operating unit type vertex in the graph

$$\mathrm{m} \subseteq \psi(\mathrm{o})$$

Solution-structure

- The structures satisfy the five axioms called combinatorially feasible solution-structures or solution-structures
- Nothing other structure is solution-structure

Set of solution-structures

- Let $S(\mathcal{P}, \mathcal{R}, \mathcal{O})$ the set of solution-structures
- The set of solution-structures are closed under union
  - The union of two solution-structures is a solution-structure

$$\sigma_1 \in S(\mathcal{P}, \mathcal{R}, \mathcal{O}) \,\&\, \sigma_2 \in S(\mathcal{P}, \mathcal{R}, \mathcal{O}) \rightarrow$$
$$\sigma_1 \cup \sigma_2 \in S(\mathcal{P}, \mathcal{R}, \mathcal{O})$$

Maximal structure

- Let $\mu(\mathcal{P}, \mathcal{R}, \mathcal{O})$ the union of all solution-structure

$$\mu(\mathcal{P}, \mathcal{R}, \mathcal{O}) = \bigcup_{\sigma \in S(\mathcal{P}, \mathcal{R}, \mathcal{O})} \sigma$$

- If the set of $S(\mathcal{P}, \mathcal{R}, \mathcal{O})$ is not empty, the $\mu(\mathcal{P}, \mathcal{R}, \mathcal{O})$ is called the maximal structure of the synthesis problem
- The maximal structure is a solution-structure
- Each solution-structure is a substructure of the maximal structure

MSG algorithm

- The MSG (Maximal Structure Generation) algorithm generates the maximal structure of a synthesis problem in polynomial time

Initialize the input of MSG

- Define the following sets
  - $\mathcal{M}$ – the set of materials
  - $\mathcal{P}$ – the set of final products
  - $\mathcal{R}$ – the set of raw materials
  - $\mathcal{O}$ – the set of plausible operating units
- The connection of the operating units through the materials defines the initial network
  - The axiom (S3) satisfies

The main steps of MSG

- The MSG algorithm consists of two main parts
  - Reduction
    - Remove materials and operating units violates axioms (S2) or (S5)
  - Composition
    - Collect the operating units which can take part of the production of a final product

Reduction part

$\mathcal{O} \coloneqq \mathcal{O} \setminus \varphi^{-}(\mathcal{R})$;

$\mathcal{M} \coloneqq \psi(\mathcal{O})$;

$\mathrm{r} \coloneqq \psi^{-}(\mathcal{O}) \setminus (\psi^{+}(\mathcal{O}) \cup \mathcal{R})$;

while $\mathrm{r} \neq \emptyset$ **do**

let $x \in \mathrm{r}$;

$\mathcal{M} \coloneqq \mathcal{M} \setminus \{x\}$;

$\mathrm{o} \coloneqq \varphi^{+}(\{x\})$;

$\mathcal{O} \coloneqq \mathcal{O} \setminus \mathrm{o}$;

$\mathrm{r} \coloneqq \left(\mathrm{r} \cup \left(\psi^{+}(\mathrm{o}) \setminus \psi^{+}(\mathcal{O})\right)\right) \setminus \{x\}$;

if $\mathcal{P} \cap \mathcal{M} \neq \mathcal{P}$ **then**

stop;

Composition part

$\mathrm{p} \coloneqq \mathcal{P}$; $\mathrm{m} \coloneqq \emptyset$; $\mathrm{o} \coloneqq \emptyset$;

while $\mathrm{p} \neq \emptyset$ **do**

let $x \in \mathrm{p}$;

$\mathrm{m} \coloneqq \mathrm{m} \cup \{x\}$;

$\mathrm{o}_x \coloneqq \varphi^{-}(\{x\})$;

$\mathrm{o} \coloneqq \mathrm{o} \cup \mathrm{o}_x$;

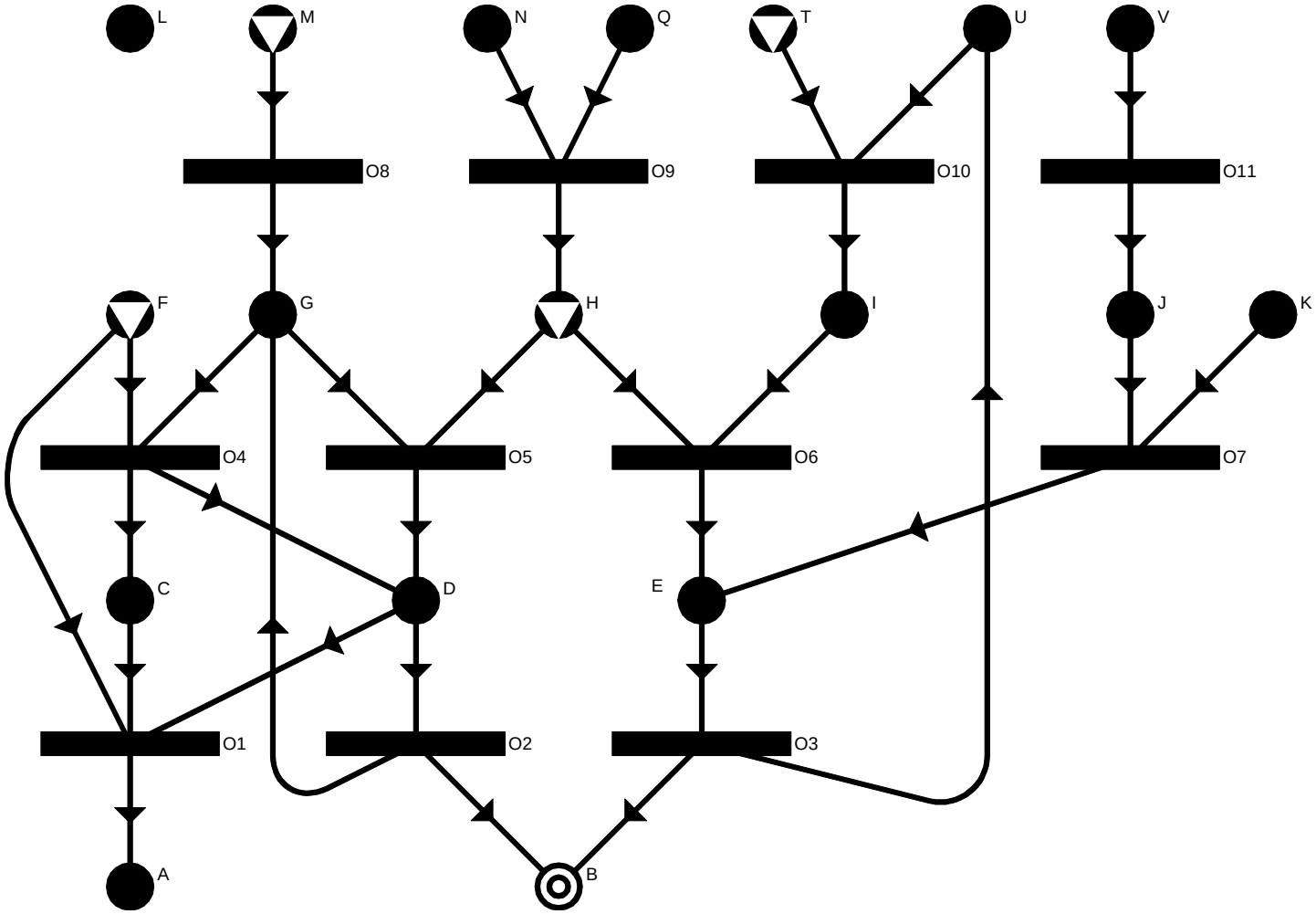
$\mathrm{p} \coloneqq \left(\mathrm{p} \cup \psi^{-}(\mathrm{o}_x)\right) \setminus (\mathcal{R} \cup \mathrm{m})$;

$\mathrm{m} \coloneqq \psi(\mathrm{o})$;

Example

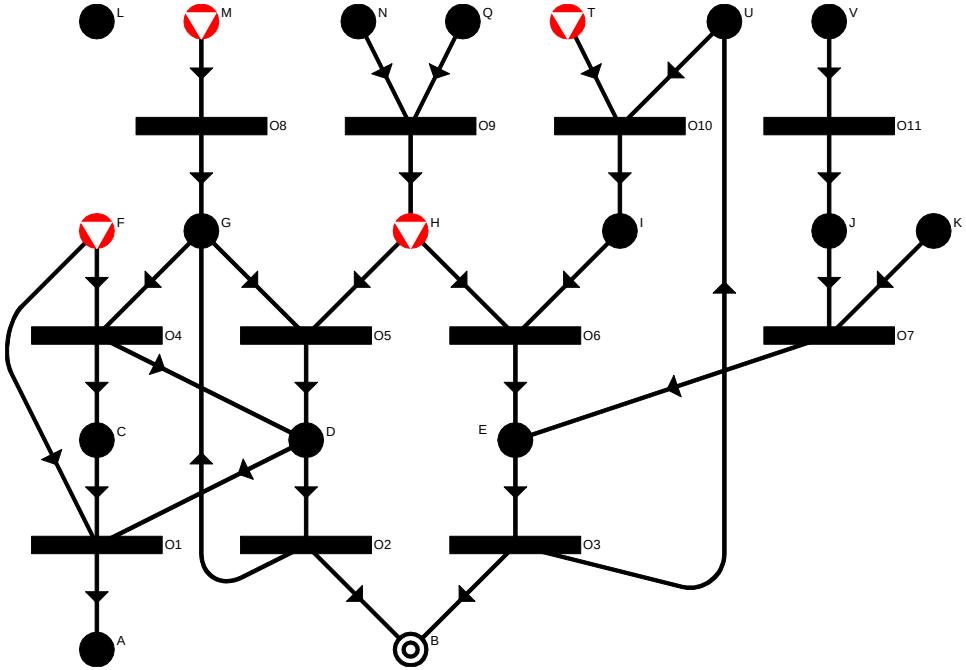
- Let the synthesis problem is the following
  - $\mathcal{M}$ = {A, B, C, D, E, F, G, H, I, J, K, L, M, N, Q, T, U, V}
  - $\mathcal{P}$ = {B}
  - $\mathcal{R}$ = {F, H, M, T}
  - $\mathcal{O}$ = {O1 = ({C, D, F}, {A}), O2 = ({D}, {B, G}), O3 = ({E}, {B, U}), O4 = ({F, G}, {C, D}), O5 = ({G, H}, {D}), O6 = ({H, I}, {E}), O7 = ({J, K}, {E}), O8 = ({M}, {G}), O9 = ({N, Q}, {H}), O10 = ({T, U}, {I}), O11 = ({V}, {J})}

Example (initial network)



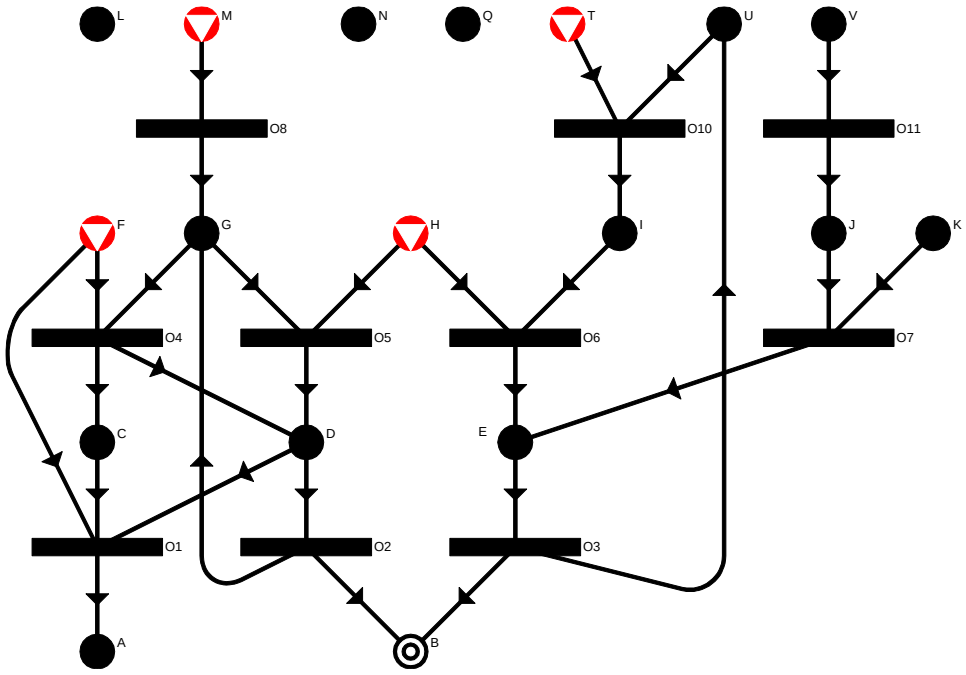
Step by step illustration

- Remove operating units producing raw materials
  - Operating unit O9 produces raw material H



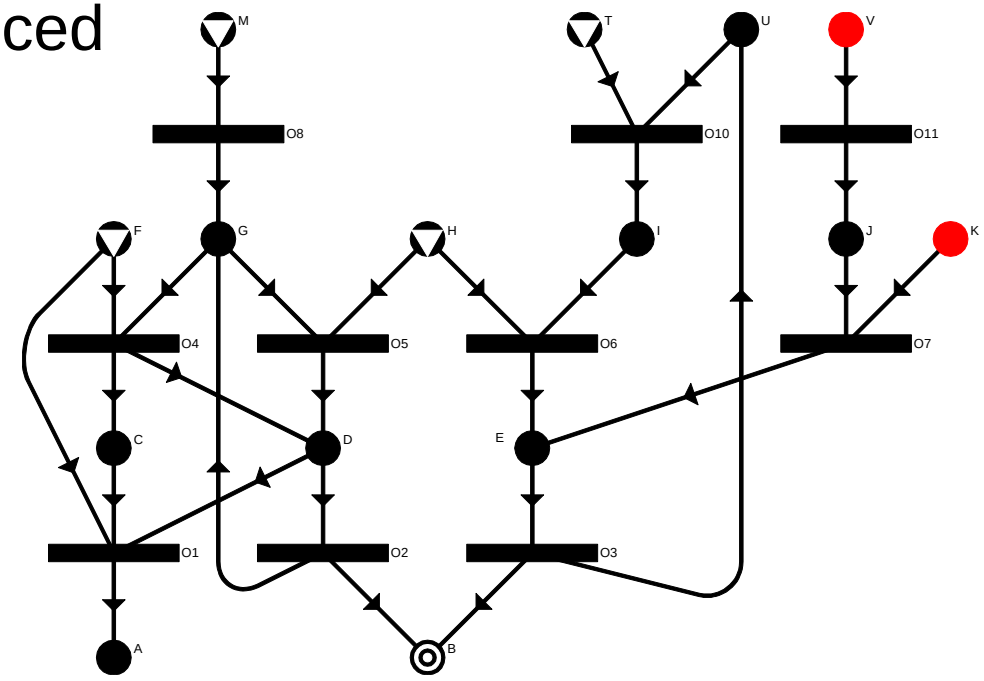
Step by step illustration

- Remove materials not produced but consumed
  - Materials L, N and Q



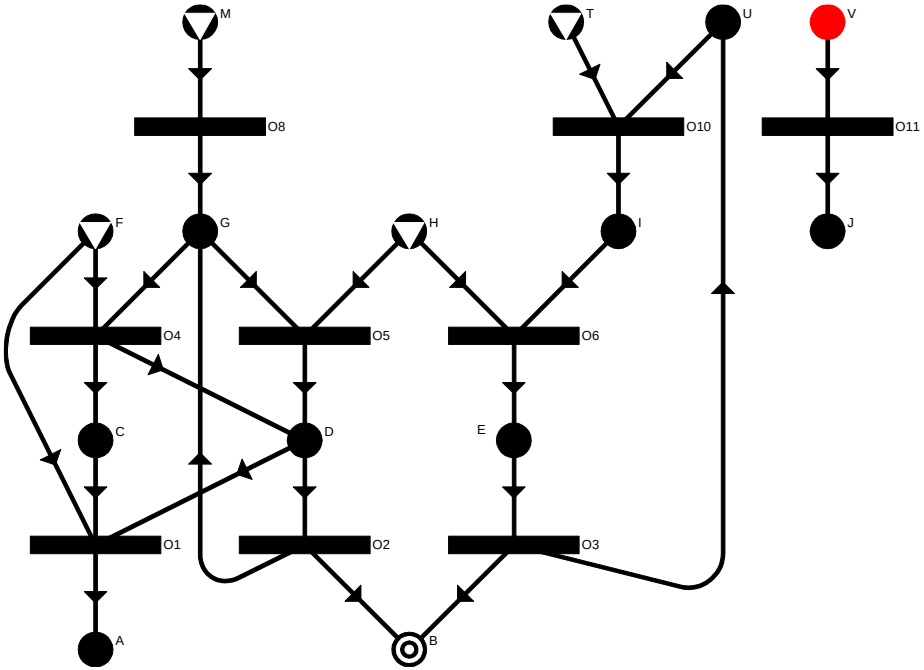
Step by step illustration

- The elements of r (denoted by red on figures) are violates axioms (S4)
  - They are consumed by operating units but not produced



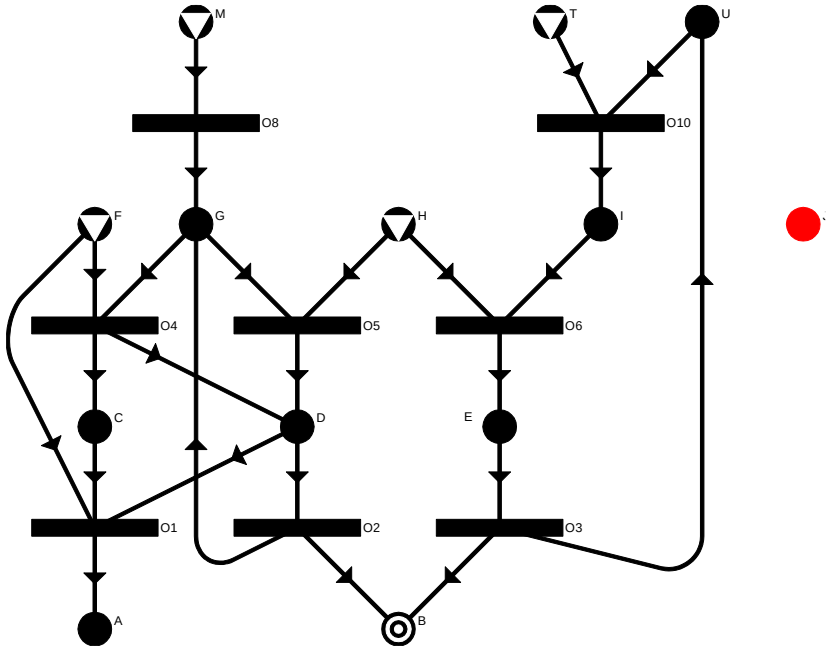
Step by step illustration

- First remove material K
  - Also remove operating unit O7 which is consuming K



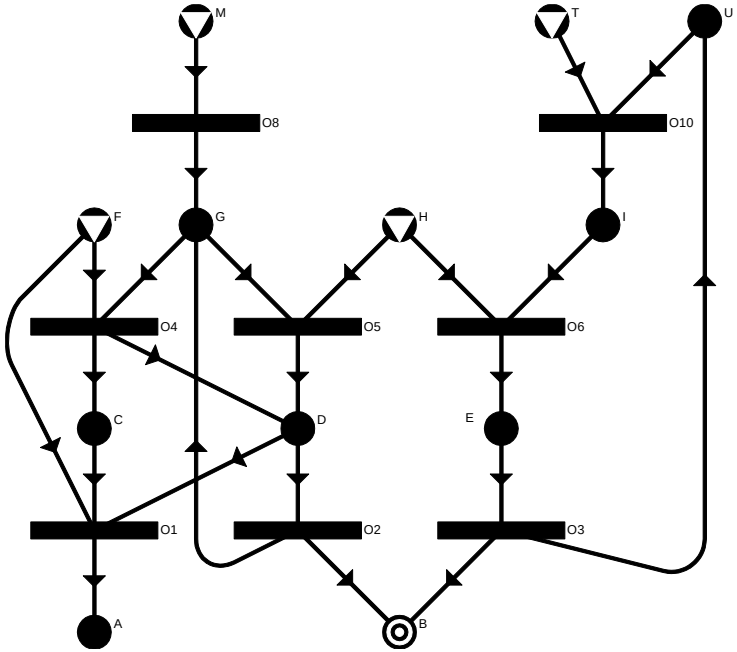
Step by step illustration

- Remove material V
  - Also remove operating unit O11
  - Put material J into r



Step by step illustration

- Remove material J
  - Set r is empty
  - Reduction part ends

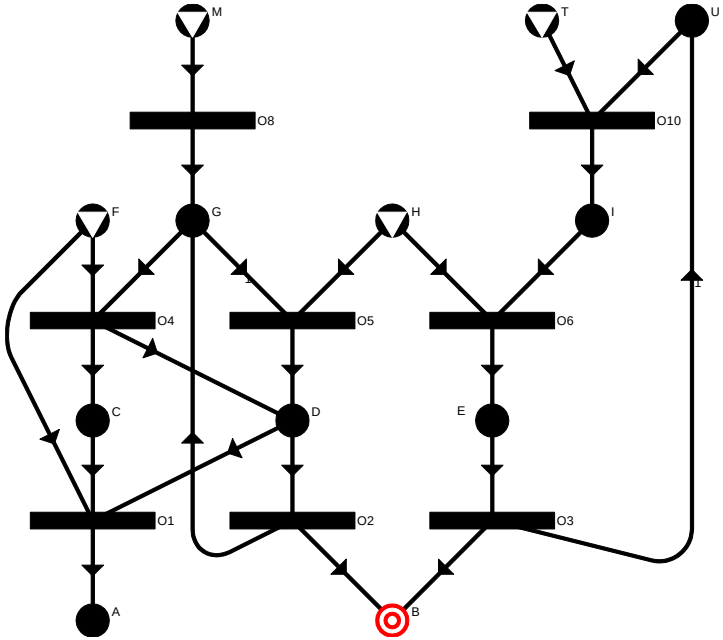


Step by step illustration

- All final product is presented in the graph
- Composition part can be started
- Notations in figures
  - p – materials have to be examined, i.e., have to be produced (denoted by red color)
  - m – examined materials, i.e. the production has been decided (denoted by blue)
  - o – operating units already included into the structure (denoted by blue)

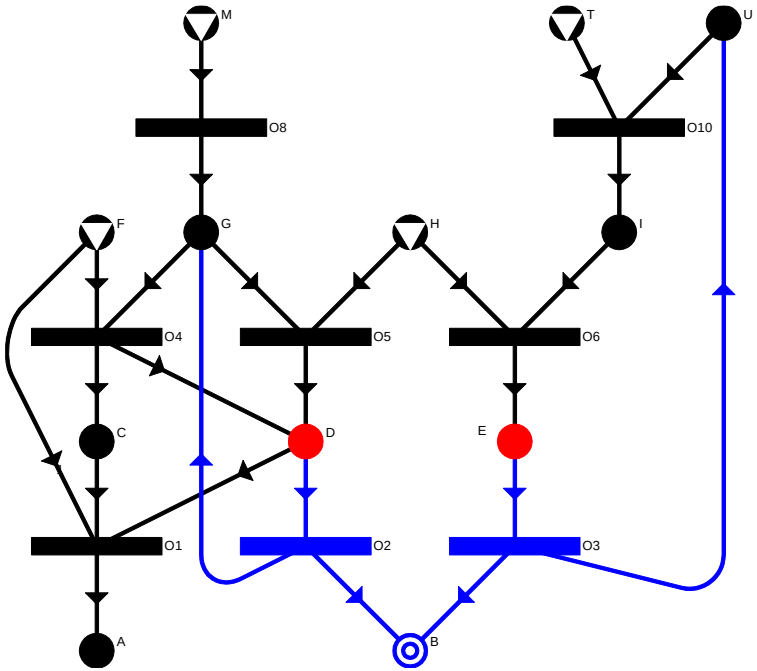
Step by step illustration

- First final product (material B) has to be produced
- Operating unit O2 and O3 can produce B



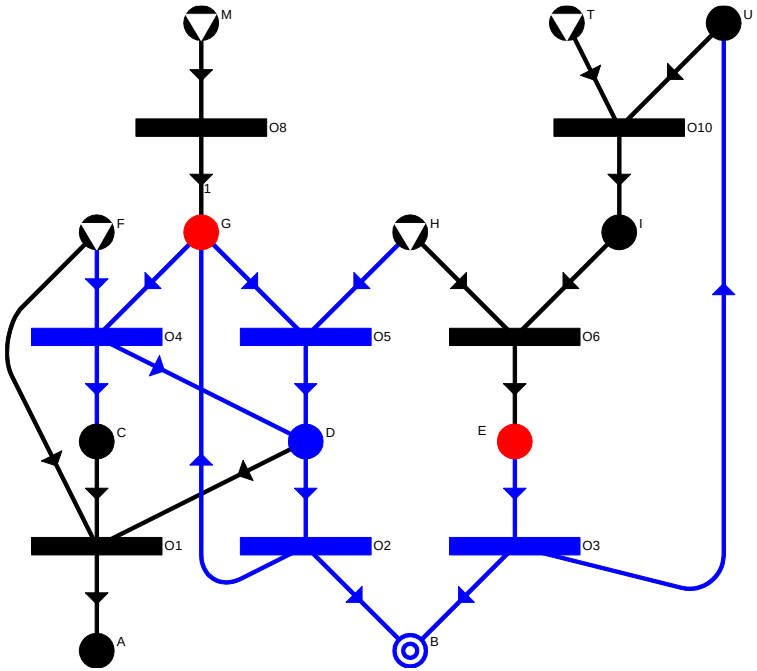
Step by step illustration

- Material D and E must be produce
- D can be produced by O4 and O5



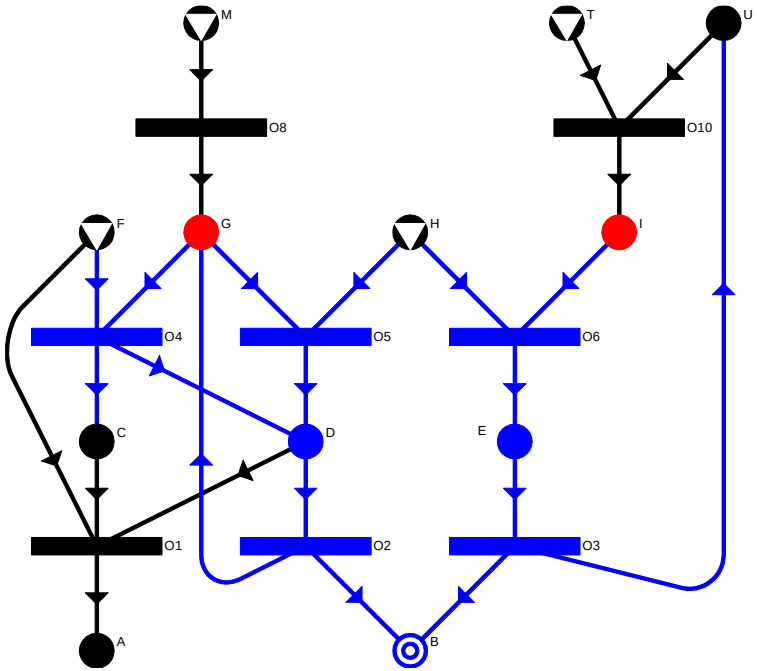
Step by step illustration

- Material E and G must be produce
- E can be produced by O6



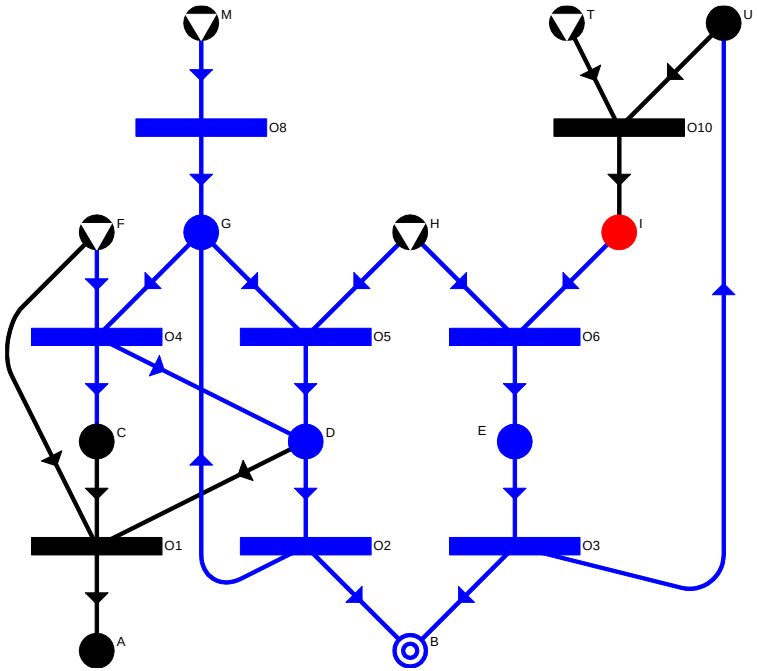
Step by step illustration

- Material G and I must be produce
- G can be produced by O8



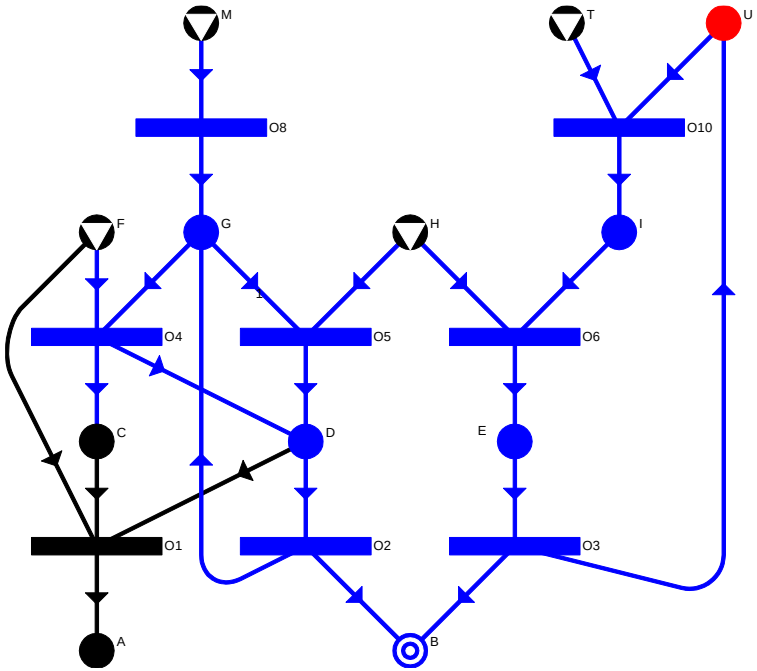
Step by step illustration

- Material I must be produce
- I can be produced by O10



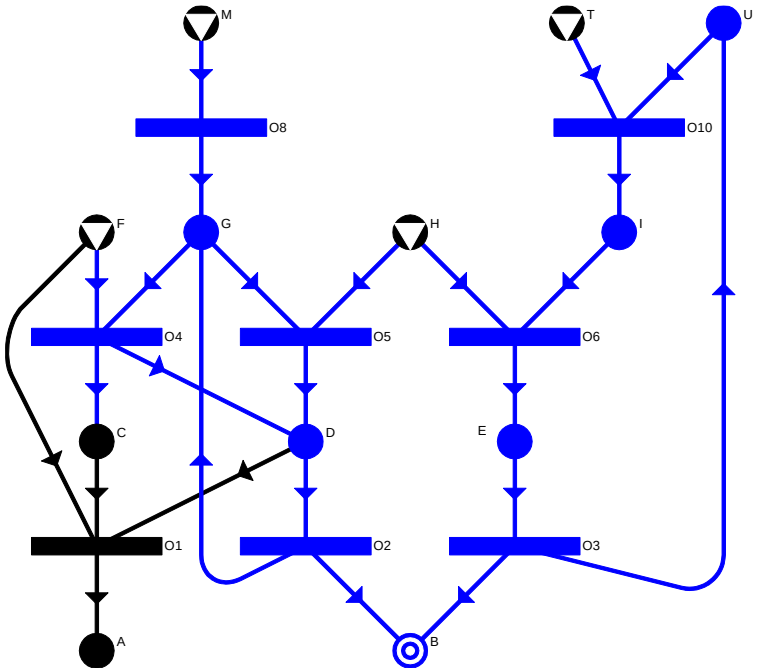
Step by step illustration

- Material U must be produce
- U can be produced by O3



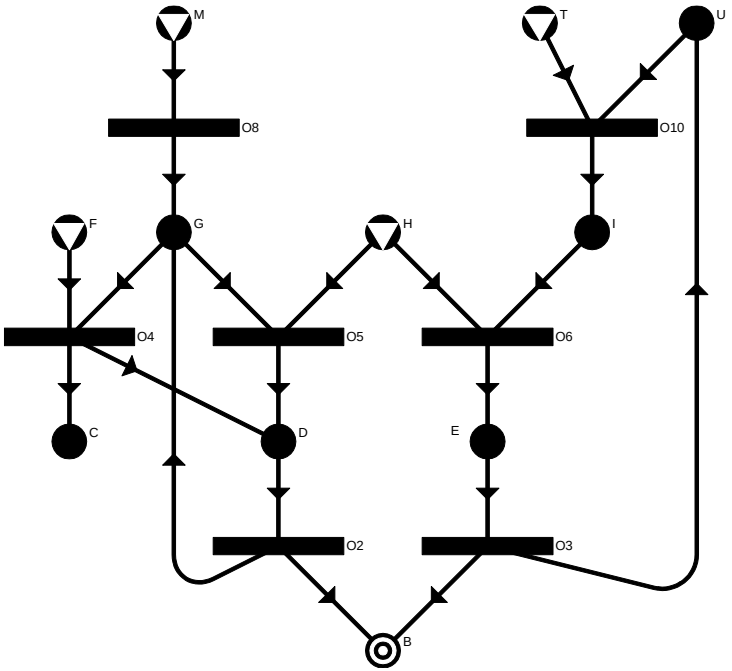
Step by step illustration

- Set p is empty
  - Operating unit O1 is not in the graph



Step by step illustration

- The set of material of the P-graph is the input and output materials of the operating units



SSG algorithm

Introduction

- Any solution structure can be optimal with appropriate parameters
- The generation of all solution structure can be useful
  - Analyzing them one by one
  - Solving the problem by exhaustive search
  - Verifying the search space reduction
  - It can provide a good basis for an efficient algorithm

Decision mapping

- Decision mapping is a tool which helps to represents the decisions during optimization and the decisions became consistent
- The decision mapping describes the decision which operating units will be used to produce a set of materials
  - Which operating units will take place in the solution-structure
- A decision is consistent if it is not inconsistent with the previous decisions

Formal description of decision mapping

- Let $\mathrm{m} \subseteq \mathcal{M}$, moreover $\forall x \in \mathrm{m}$, $\delta(x) \subseteq \varphi^{-}(x)$ and $\Delta(x) = \varphi^{-}(x)$
- $\Delta[\mathrm{m}] = \{(x, \Delta(x)) | x \in \mathrm{m}\}$ is a mapping over m
- $\delta[\mathrm{m}] = \{(x, \delta(x)) | x \in \mathrm{m}\}$ is a decision mapping over m
- The complement of decision mapping δ over m is $\bar{\delta}[\mathrm{m}] = \{(x, \mathrm{y}) | x \in \mathrm{m}, \mathrm{y} = \varphi^{-}(x) \setminus \delta(x)\}$

Decision mapping

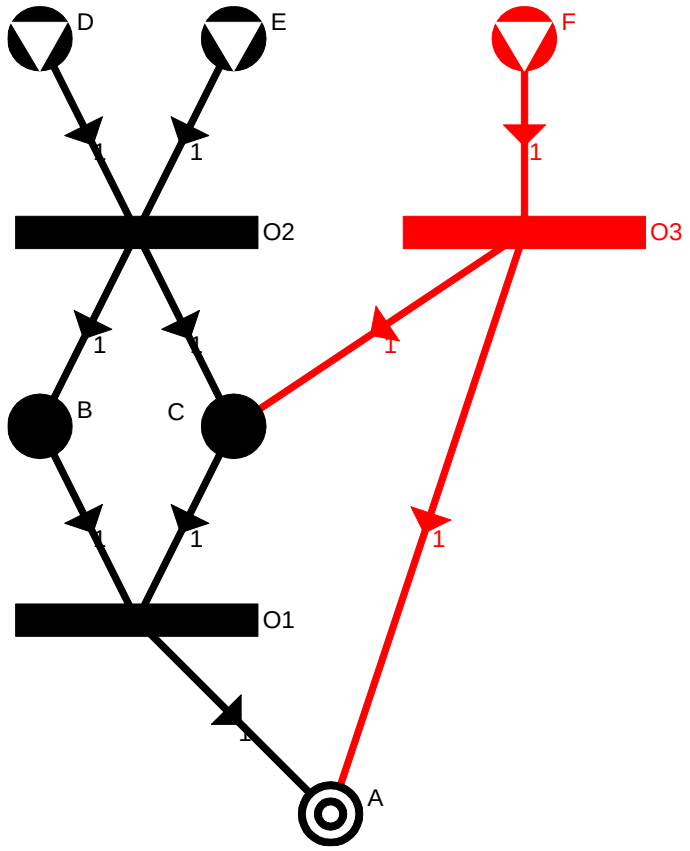
- The mapping is a set of pairs, where the first element of a pair is a material, the second element of the pair is the set of operating units which can produce the material
- In case of decision mapping the second element of the pair is the set of material which are chosen to produce the material
- In case of complement the second element of the pair is the set of material which are excluded from the structure

Decision mapping

- If there are parenthesis after δ or Δ, it means the parameter is a material and the result is a set of operating units
- If there are brackets after δ or Δ, it means the parameter is a set of materials and the result is a set of pairs (defined above)

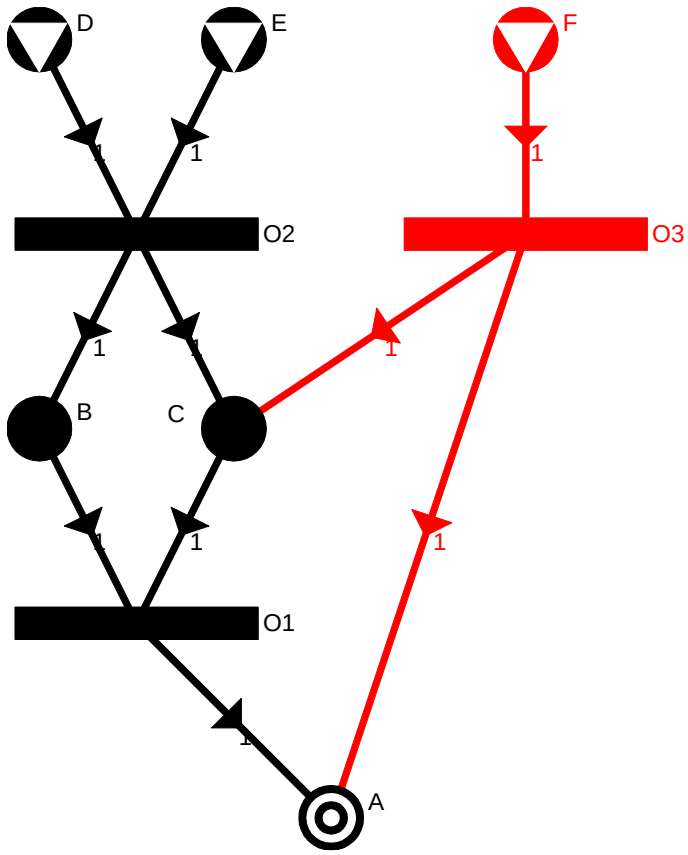
Example

- Operating units O1 and O2 are included in the structure, O3 is excluded from the structure
  - $\Delta(A) = \{O1, O3\}$, $\delta(A) = \{O1\}$
  - $\Delta(B) = \{O2\}$, $\delta(B) = \{O2\}$,
  - $\Delta(C) = \{O2, O3\}$, $\delta(C) = \{O2\}$
  - $\Delta(D) = \Delta(E) = \Delta(F) = \emptyset$, $\delta(D) = \delta(E) = \delta(F) = \emptyset$



Example

- Decision mapping can be defined for any set of materials
  - Δ [{A, B, C, D, E, F}] = {(A, Δ(A)), (B, Δ(B)), (C, Δ(C)), (D, Δ(D)), (E, Δ(E)), (F, Δ(F))} = {(A, {O1, O3}), (B, {O2}), (C, {O2, O3}), (D, ∅), (E, ∅), (F, ∅)}
  - δ[{A, B, C}] = {(A, δ(A)), (B, δ(B)), (C, δ(C))} = {(A, {O1}), (B, {O2}), (C, {O2})}



Consistent decision mapping

- Decision mapping $\delta[\mathrm{m}]$ is consistent if
  - $|\mathrm{m}| \leq 1$ or
  - $\left(\delta(x) \cap \delta(y)\right) \cup \bar{\delta}(x) \cap \bar{\delta}(y) = \Delta(x) \cap \Delta(y)$ for all $x, y \in \mathrm{m}$

Variables in SSG algorithm

- p – materials have to be examined, i.e., have to be produced
- m – examined materials, i.e., the production has been decided
- $\delta[\mathrm{m}]$ – decision mapping representing the previous decisions
- x – the material chosen for decision
- C – the set of possible decisions about material x
- c – the current decision (set of operating units)

Initialization of the SSG algorithm

begin

if $\mathcal{P} = \emptyset$ **then**

stop;

SSG($\mathcal{P}$, $\emptyset$, $\emptyset$);

end

SSG algorithm

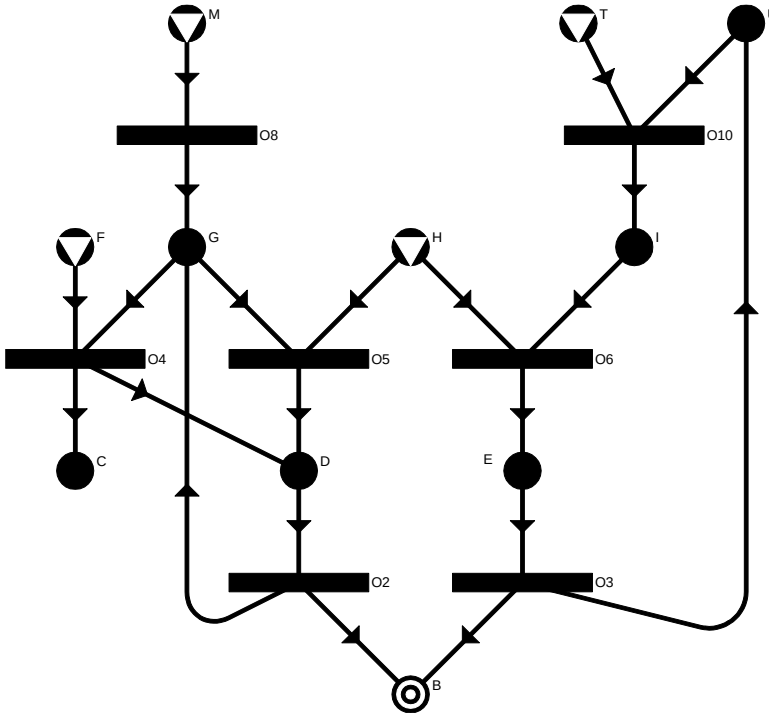
procedure SSG(p, m, δ[p])
begin
  if $\mathrm{p} = \emptyset$ **then**
    print $\delta[\mathrm{m}]$;
    return;
  let $x \in \mathrm{p}$;
  $\mathrm{C} \coloneqq \wp\big(\Delta(x)\big) \setminus \{\emptyset\}$;
  for all $\mathrm{c} \in \mathrm{C}$ **do**
    if $\forall y \in \mathrm{m}, \mathrm{c} \cap \bar{\delta}(y) = \emptyset$ **and** $\Delta(x) \setminus \mathrm{c} \cap \delta(y) = \emptyset$ **then**
      $\mathrm{SSG}(\mathrm{p} \cup \psi^{-}(\mathrm{c}) \setminus (\mathcal{R} \cup \mathrm{m} \cup \{x\}), \mathrm{m} \cup \{x\}, \delta[\mathrm{m}] \cup \{(x, \mathrm{c})\})$;
end

SSG

- The algorithm has same similarity with the composition part of the MSG algorithm
- It starts from the products and make decisions about the productions
- Multiple decisions available → search tree represents the work of the algorithm

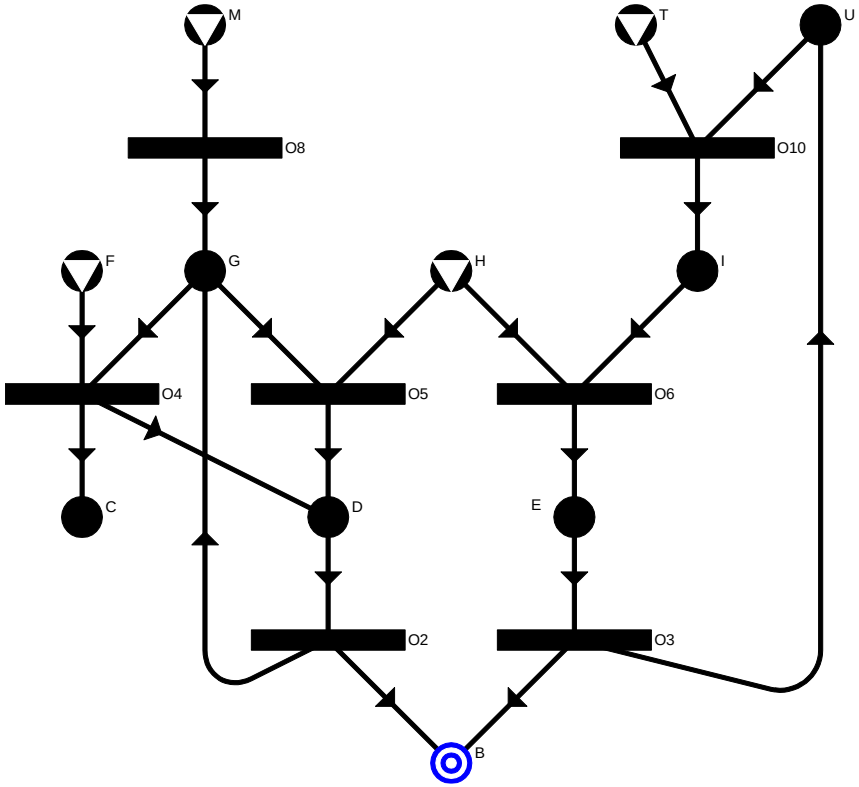
Example

- Continue the example from the previous section, where maximal structure has been generated



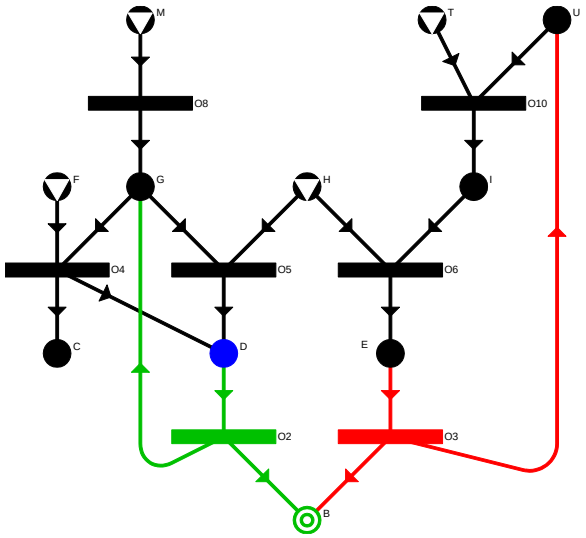
Initialization

- Initialization defines the materials have to be produced

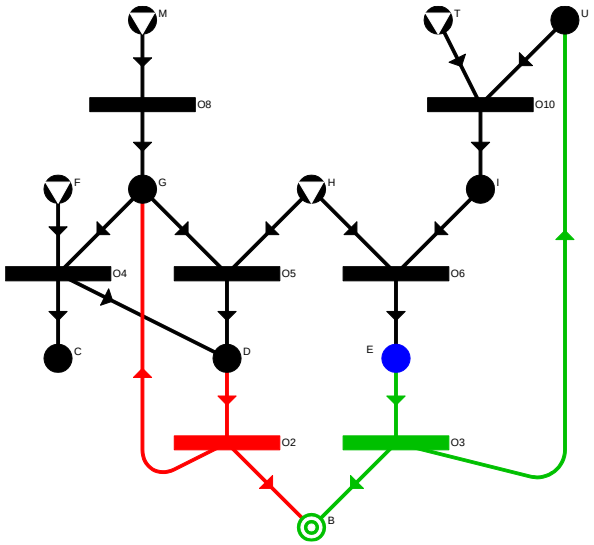


First step

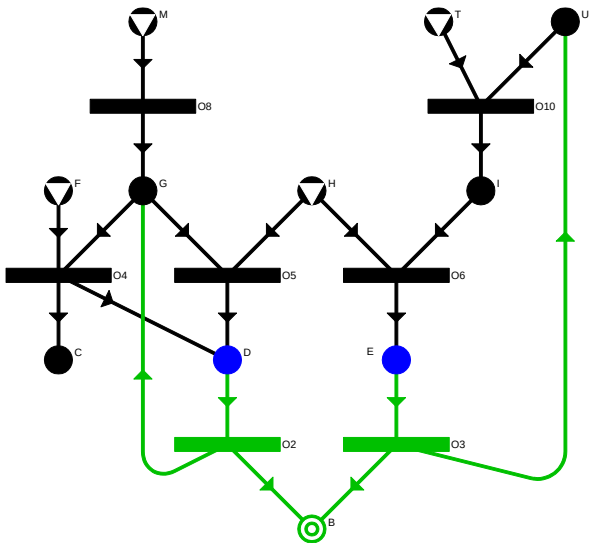
- 3 possible decisions



Producing by O2



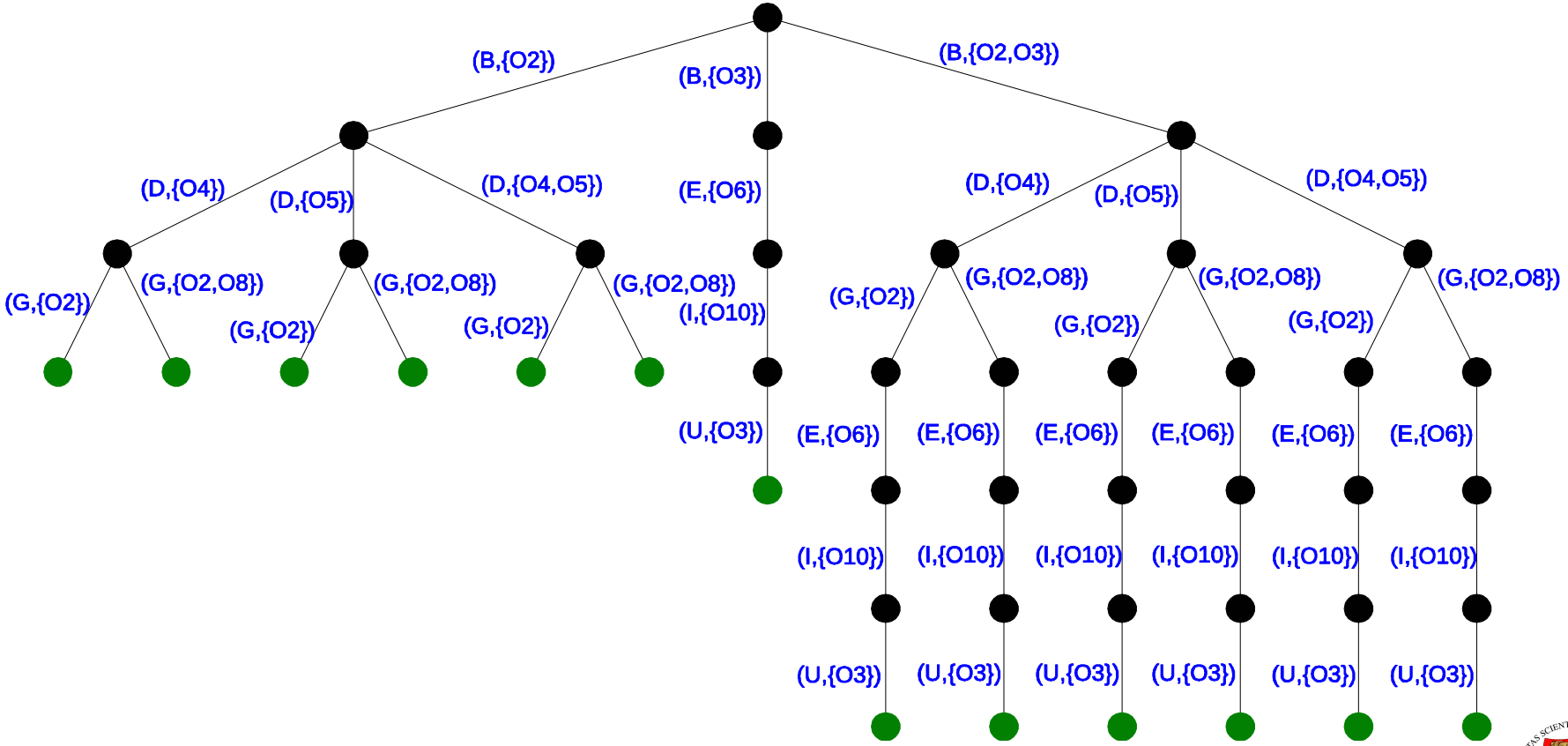
Producing by O3



Producing by O2 and O3

Search tree

- The work of the SSG algorithm



Result

- There are 13 solution-structures

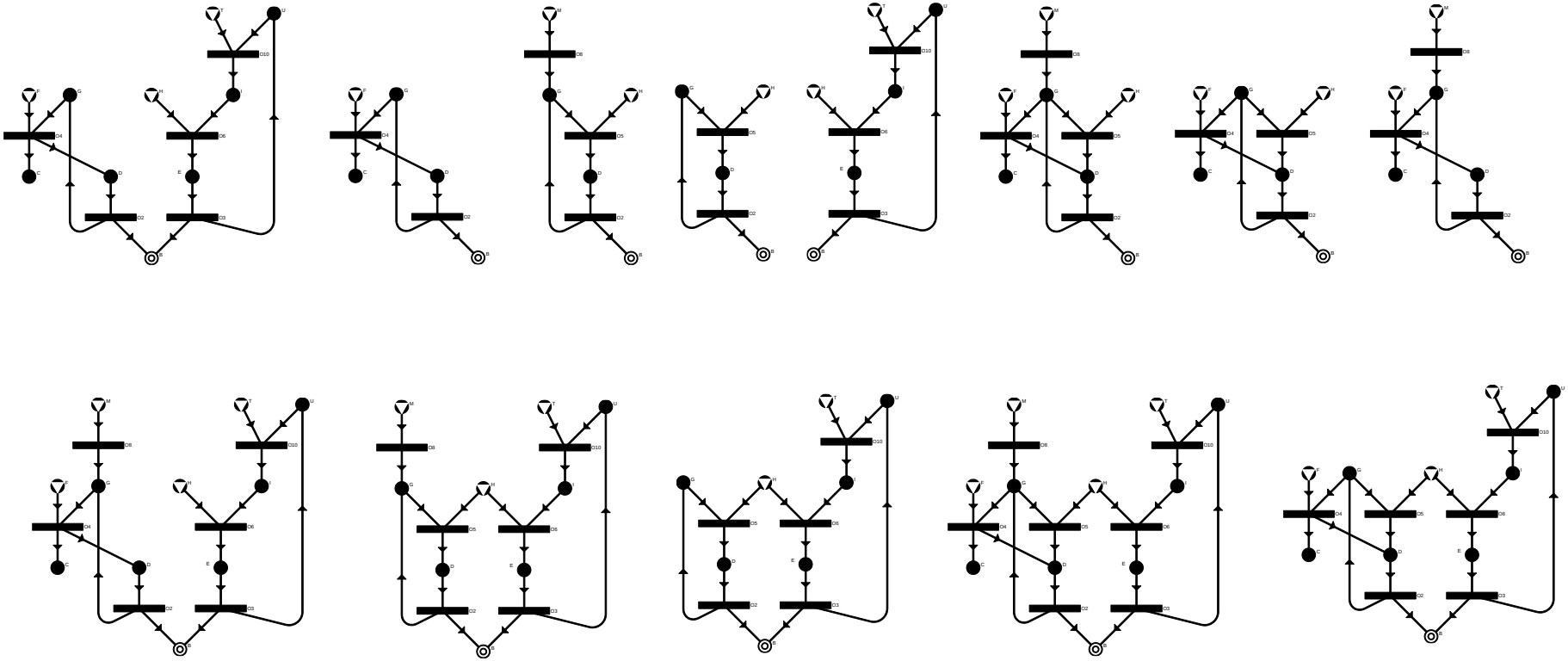


ABB algorithm

Introduction

- MSG and SSG take into account only structural informations
- Parameters are also important
- Mathematical model is needed
  - MILP model

Mathematical model

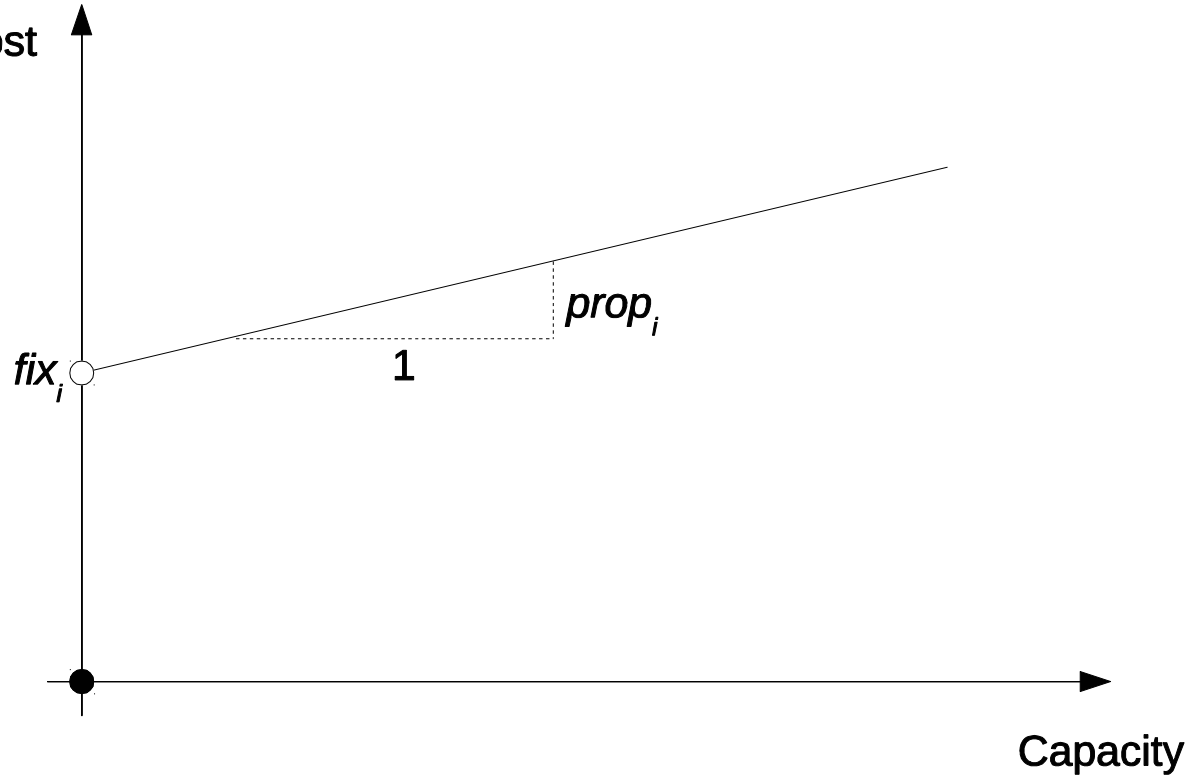
- Aim is to minimize the overall cost
  - Investment cost
  - Operational cost
  - Material cost
- Constraints
  - Lower bounds on the amounts of products to be manufactured to meet the demand
  - Availability of raw materials
  - Mass balance

Decision variables

- Two variables for each operating unit
  - y_i denotes the existence of operating unit $o_i \in \mathcal{O}$ in the solution
  - x_i denotes the capacity of operating unit $o_i \in \mathcal{O}$ in the solution

Cost of operating units

- Investment and operational cost are similar, we do not distinguish them



Cost of operating units

- The cost of operating unit o_i

$$y_i(fix_i + prop_i x_i)$$

- Linearization
  - The cost

$$fix_i y_i + prop_i x_i$$

  - Additional constraint

$$x_i \leq cap_i y_i$$

    - Where cap_i is the maximum capacity of o_i

Cost of raw materials

- The overall consumption of raw material m_j

$$\sum_{o_i \in \varphi^+(j)} x_i ir_{ij}$$

  - Where ir_{ij} is the consumption rate of m_j by $o_i \in \mathcal{O}$

- The cost of raw material r_j

$$price_j \sum_{o_i \in \varphi^+(j)} x_i \, ir_{ij}$$

  - Where $price_j$ is the price of raw material $r_j \in \mathcal{R}$

Objective function

$$\min \sum_{o_i \in \mathcal{O}} (fix_i y_i + prop_i x_i)$$

$$+ \sum_{r_j \in \mathcal{R}} \left(price_j \sum_{o_i \in \varphi^+(j)} x_i \, ir_{ij} \right)$$

Constraints

- Constraint from linearization of objective function

$$x_i \leq cap_i y_i \quad o_i \in \mathcal{O}$$

Constraints

- Availability of raw materials

$$\sum_{o_i \in \varphi^+(j)} x_i ir_{ij} \leq max_j \quad m_j \in \mathcal{R}$$

Constraints

- Lower bounds on the amounts of products to be manufactured to meet the demand
- Product can be produced and purchased

$$\sum_{o_i \in \varphi^-(j)} x_i or_{ij} - \sum_{o_i \in \varphi^+(j)} x_i ir_{ij} \geq min_j \quad m_j \in \mathcal{P}$$

  - Where or_{ij} is the production rate of m_j by $o_i \in \mathcal{O}$

Constraints

- Mass balance

$$\sum_{o_i \in \varphi^-(j)} x_i or_{ij} \geq \sum_{o_i \in \varphi^+(j)} x_i ir_{ij}$$

$$m_j \in \mathcal{M} \setminus (\mathcal{R} \cup \mathcal{P})$$

Constraints

- The last three constraints can be merged

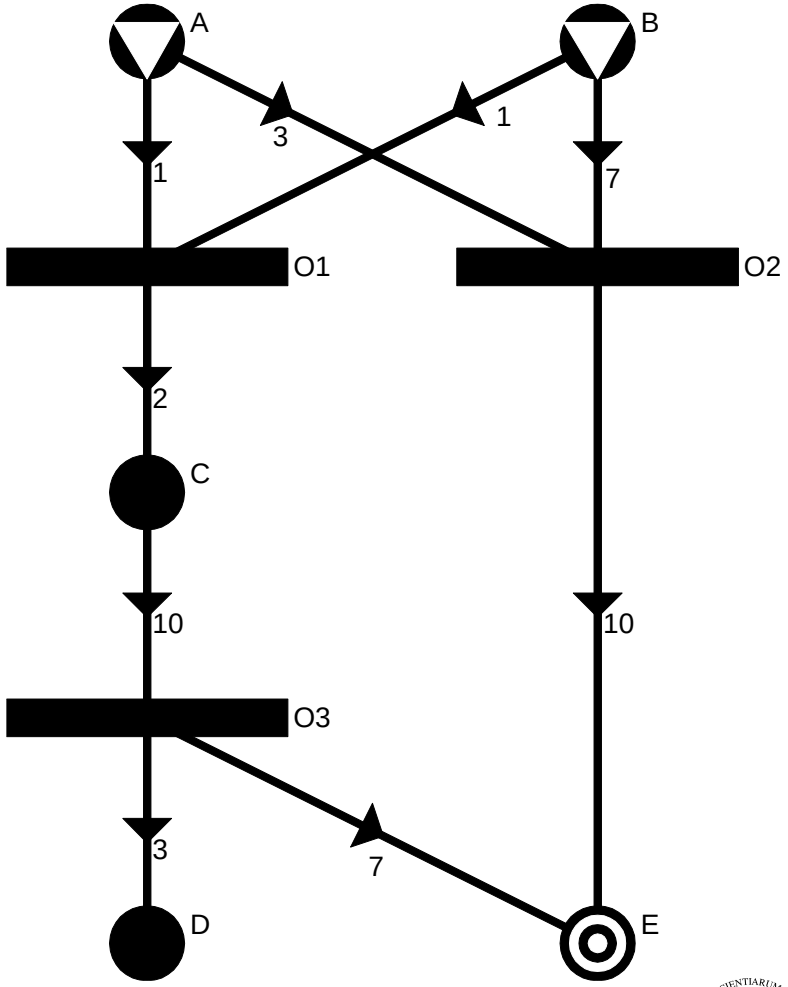
$$l_j \leq \sum_{o_i \in \varphi^-(j)} x_i or_{ij} - \sum_{o_i \in \varphi^+(j)} x_i ir_{ij} \leq u_j$$

$$m_j \in \mathcal{M}$$

  - Where
    - $l_j = -max_j$ and $u_j = 0$ if $m_j \in \mathcal{R}$
    - $l_j = min_j$ and u_j is an arbitrary big number if $m_j \in \mathcal{P}$
    - $l_j = 0$ and u_j represents the remaining amount of material m_j if $m_j \in \mathcal{M} \setminus (\mathcal{R} \cup \mathcal{P})$

Example

- Define the MILP model of following the maximal structure
- The input and output ratios are given as weights of the arcs



Example (parameters)

Operating unit	cap	fix	$prop$
O1	10	5	5
O2	10	10	10
O3	10	5	5

Materials	min	max	$price$
A		10	2
B		20	1.5
E	5		

Example (model)

$$\min 5y_1 + 8.5x_1 + 10y_2 + 26.5x_2 + 5y_3 + 5x_3$$

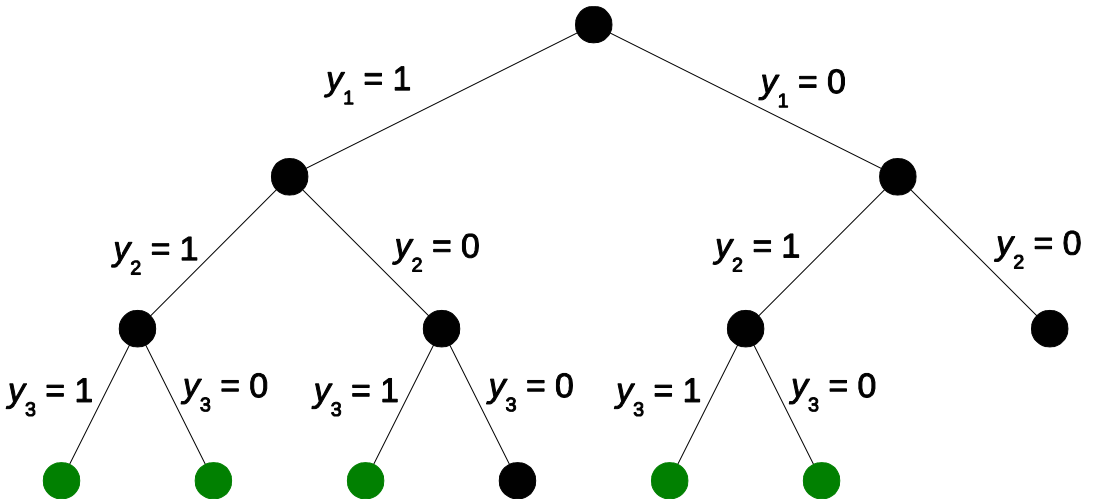
s.t.

$$\begin{array}{rcccl}
0 & \leq & x_1 & \leq & 10y_2 \\
0 & \leq & x_2 & \leq & 10y_2 \\
0 & \leq & x_3 & \leq & 10y_3 \\
-10 & \leq & -x_1 - 3x_2 & \leq & \infty \\
-20 & \leq & -x_1 - 7x_2 & \leq & \infty \\
0 & \leq & 2x_1 - 10x_3 & \leq & \infty \\
0 & \leq & 3x_3 & \leq & \infty \\
5 & \leq & 10x_2 + 7x_3 & \leq & \infty
\end{array}$$

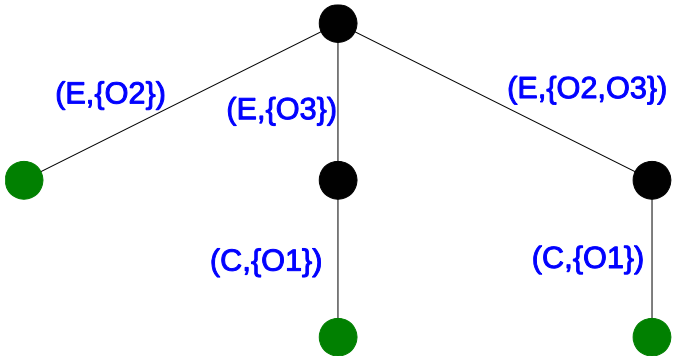
Solving the model

- General MILP solver
- SSG algorithm and LP solver for each solution structure
- General branch and bound method
- ABB

Search tree



General branch and bound



SSG

ABB

- Based on SSG
  - Same search tree
- Lower bound generated by solving the relaxed MILP
- Input – maximal structure
- Output – optimal structure

New variables in the ABB algorithm

- U – value of the current best solution
- $current\_best$ – the current best solution
- $bound$ – lower bound of the subproblem

Initialization of the ABB algorithm

begin

$U = \infty$;

ABB($\mathcal{P}$, $\emptyset$, $\emptyset$);

if $U < \infty$ **then**

print $current\_best$;

else

print “There is no solution”;

end

ABB algorithm

procedure ABB(p, m, δ[p])

begin

let $x \in$ p;

C $\coloneqq \wp\big(\Delta(x)\big) \setminus \{\emptyset\}$;

for all c $\in$ C **do**

if $\forall y \in$ m, c $\cap\ \bar{\delta}(y) = \emptyset$ **and** $(\Delta(x) \setminus$ c$) \cap \delta(y) = \emptyset$ **then**

p′ $= \big($p $\cup\ \psi^{-}($c$)\big) \setminus (\mathcal{R} \cup$ m $\cup \{x\})$;

m′ $=$ m $\cup \{x\}$;

δ[m′] $=\ \delta$[m] $\cup \{(x,$ c$)\}$;

$bound$ = BOUND(δ[m′]);

if $U \geq bound$ **then**

if p′ $= \emptyset$ **then**

$U = bound$;

$currentbest = \delta$[m′];

else

ABB(p′, m′, δ[m′]);

end

Neutral extension

- $\delta_x[\mathrm{m} \cup \{x\}] = \delta[\mathrm{m}] \cup \{(x, \mathrm{d})\}$ is a **direct neutral extension** of $\delta[\mathrm{m}]$ consistent decision mapping, if $x \in \left(\psi^{-}\left(\varphi(\delta[\mathrm{m}])\right) \cup \mathcal{P}\right) \setminus (\mathrm{m} \cup \mathcal{R})$, $\mathrm{d} \subset \Delta[\mathrm{m}]$, and c is inconsistent if $\mathrm{c} \in \wp\left(\delta(x)\right) \setminus \{\emptyset, \mathrm{d}\}$
  - I.e., only d is consistent decision
- $\delta_n[\mathrm{m}_n]$ $(n = 0,1,\ldots)$ decision mapping is a **neutral extension** of $\delta_0[\mathrm{m}_0]$ if there exists $\delta_0[\mathrm{m}_0]$, $\delta_1[\mathrm{m}_1]$, …, $\delta_n[\mathrm{m}_n]$ such that $\delta_i[\mathrm{m}_i]$ is direct neutral extension of $\delta_{i-1}[\mathrm{m}_{i-1}]$ $(i = 1,2,\ldots,n)$
- $\hat{\delta}[\hat{\mathrm{m}}]$ consistent decision mapping is the **maximal neutral extension** of $\delta[\mathrm{m}]$, if it is neutral extension of $\delta[\mathrm{m}]$ and it has no neutral extension

Initialization of the extended ABB algorithm

begin

$U = \infty$;

let $\hat{\delta}[\hat{\mathrm{m}}]$ the maximal neutral extension of $\delta[\mathrm{m}]$;

$\mathrm{p} = \left(\psi^{-}\left(\varphi\left(\hat{\delta}[\hat{\mathrm{m}}]\right)\right)\right) \cup \mathcal{P} \setminus (\hat{\mathrm{m}} \cup \mathcal{R})$;

if $\mathrm{p} = \emptyset$ **then**

U = BOUND($\hat{\delta}[\hat{\mathrm{m}}]$);

update $currentbest$;

else

ABB(p, $\hat{\mathrm{m}}$, $\hat{\delta}[\hat{\mathrm{m}}]$);

if $U < \infty$ **then**

print $current\_best$;

else

print “There is no solution”;

end

Extended ABB algorithm

procedure ABB(p, m, δ[p])

begin

let $x \in$ p;

C $\coloneqq \wp(\Delta(x)) \setminus \{\emptyset\}$;

for all c $\in$ C **do**

if $\forall y \in$ m, c $\cap \bar{\delta}(y) = \emptyset$ **and** $(\Delta(x) \setminus$ c$) \cap \delta(y) = \emptyset$ **then**

p′ $= \left(\psi^{-}\left(\varphi\left(\hat{\delta}[\hat{\mathfrak{m}}']\right)\right)\right) \cup \mathcal{P} \setminus (\hat{\mathfrak{m}}' \cup \mathcal{R})$;

m′ $=$ m $\cup \{x\}$;

δ[m′] $=$ δ[m] $\cup \{(x,$ c$)\}$;

let $\hat{\delta}[\hat{\mathfrak{m}}']$ the maximal neutral extension of $\delta[m']$;

$bound$ = BOUND($\hat{\delta}[\hat{\mathfrak{m}}']$);

if $U \geq bound$ **then**

if p′ $= \emptyset$ **then**

$U = bound$;

$currentbest = \hat{\delta}[\hat{\mathfrak{m}}']$;

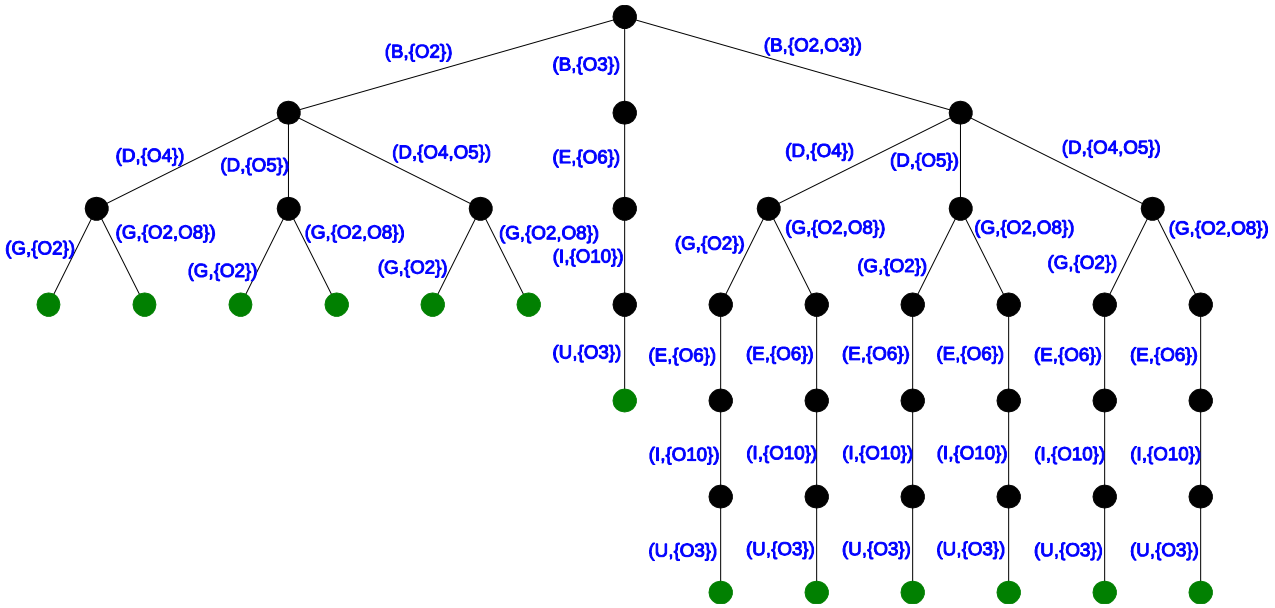
else

ABB(p′, $\hat{\mathfrak{m}}'$, $\hat{\delta}[\hat{\mathfrak{m}}']$);

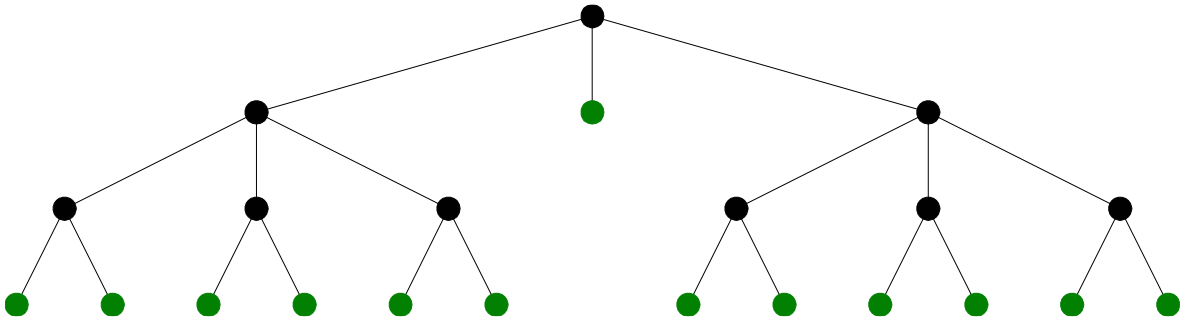
end

Acceleration

Search tree without neutral extension



Search tree with neutral extension



Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: axioms and theorems. Chem. Eng. Sci. 1992, 47, 1973.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Combinatorial algorithms for process synthesis. Comput. Chem. Eng. 1992, 16, S313.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- http://www.p-graph.com

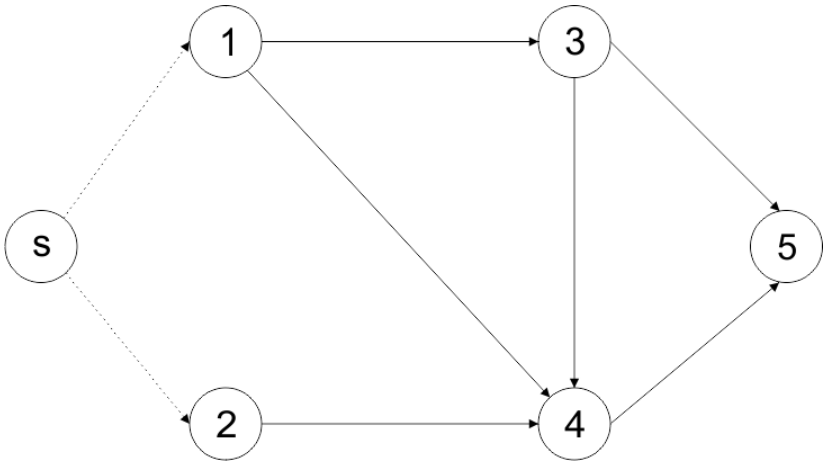
Synthesis of Reliable Process Networks

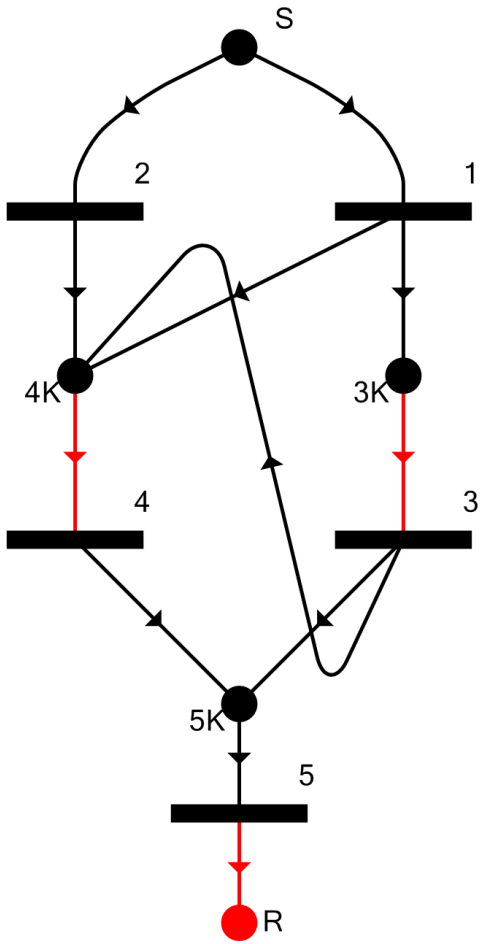
Table of Contents

Introduciton

- The system reliability, or in other words the probability of an error-free operation, has been a research area for long time (Neumann, 1956).
- In the conventional approach there are some points which is not clear enough (rigorously), i.e.
  - the concept and structure of the system model
  - the definition of operability
- To overcome the difficulties and to reach new results, application of the P-graph methodology for the studying of the process networks is proposed.
- Hereinafter, the conventional system model is generalized.

Conventional model and P-graph model





The concept of the network model

- A network is built up of operating units.
- The structure of the network is described by the system model.
- An operating unit is either working, or not.

$$x_i = \begin{cases} 1 & \text{if the i operating unit is working} \\ 0 & \text{if the i operating unit is not working} \end{cases}$$

$X = (x_1, x_2, \ldots x_n)$ a state of the system

- The operability of the system is a function of states:

$$\exists \Psi(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if the system is working} \\ 0 & \text{if the system is not working} \end{cases}$$

- Note that it is a monotone logic function.

Definition of reliability

- The space describing the possibly states of the system

$$\Omega = \{0,1\}^n = \{\ (0,0,...,0),(0,0,...,1),...,(1,1,...,1)\ \}$$

- The state of the system, an elementary event

$$(b_1, b_2, ..., b_n) \in \Omega$$

- A system event

$$E = \{(x_1, x_2, ..., x_n) \in \Omega : \varphi(x_1, x_2, ..., x_n) = 1\}$$

- Let *O* be the „system is working” system event

$$O = \{(x_1, x_2, ..., x_n) \in \Omega : \Psi(x_1, x_2, ..., x_n) = 1\}$$

- The Reliability of the system = the probability of the event *O*

$$R = P(\ O\ )$$

Probability of system events

- Let the probability of the states of the operating units be

$$p_i = P(x_i = 1) \qquad 1 - p_i = P(x_i = 0)$$

- For the elementery event $(b_1, b_2, \ldots, b_n) \in \Omega$
- the probability of its supervention is

$$P((x_1, x_2, \ldots, x_n) = (b_1, b_2, \ldots, b_n)) = \prod_{i=1}^{n} p_i^{b_i} (1 - p_i)^{(1-b_i)}$$

- The probability of a system event is given by the

$$P(A) = \sum_{(x_1, x_2, \ldots, x_n) \in A} \prod_{i=1}^{n} p_i^{x_i} (1 - p_i)^{(1-x_i)}$$

- polynom.
- Thus, there exists a polynom for reliability as well

$$R = P(\Psi(x_1, x_2, \ldots, x_n) = 1) = Q(p_1, p_2, \ldots, p_n)$$

Characteristic polynoms

- An arbitrary E system event

$$E = \{(x_1, x_2, ..., x_n) \in \Omega : \varphi(x_1, x_2, ..., x_n) = 1\}$$

- Characteristic polynom is that Q polynom, for which

$$Q(x_1, x_2, ..., x_n) = 1 \quad ha \quad (x_1, x_2, ..., x_n) \in E$$

$$Q(x_1, x_2, ..., x_n) = 0 \quad ha \quad (x_1, x_2, ..., x_n) \notin E$$

and

$$Q(p_1, p_2, ..., p_n) = P(E) \quad ha$$

$$p_1 = P(x_1 = 1), p_2 = P(x_2 = 1), ..., p_n = P(x_n = 1)$$

Characteristic polynoms (2)

- Let us consider the following system event

$$\varphi(x_1, x_2, x_3) = x_2 \vee x_1 \wedge \overline{x}_2 \wedge x_3$$

$$E = \{(x_1, x_2, x_3) \in \Omega : \varphi(x_1, x_2, x_3) = 1\}$$

- Then

$$Q(x_1, x_2, x_3) = x_2 + x_1 * (1 - x_2) * x_3, \quad x_1, x_2, x_3 \in [0,1]$$

- Why?
- Because the system events defined by the clouses of the logical function are disjoint (+) and the system events defined by variables that build up the clouses are independent (*).

Two specific characteristic polynoms

- The probability of the i-th operating unit is

$$p_i = P(x_i = 1) \qquad 1 - p_i = P(x_i = 0)$$

  - Serial system $\Psi(x_1, x_2, ..., x_n) = x_1 \wedge x_2 \wedge ... \wedge x_n$

$$R = P(\Psi(x_1, x_2, ..., x_n) = 1) = \prod_{i=1}^{n} p_i$$

$$Q(x_1, x_2, ..., x_n) = x_1 * x_2 * ... * x_n$$

  - Parallel system $\Psi(x_1, x_2, ..., x_n) = x_1 \vee x_2 \vee ... \vee x_n$

$$R = P(\Psi(x_1, x_2, ..., x_n) = 1) = 1 - \prod_{i=1}^{n} (1 - p_i)$$

$$Q(x_1, x_2, ..., x_n) = 1 - (1 - x_1) * (1 - x_2) * ... * (1 - x_n)$$

Reliability of process networks

- A process network is considered working, if the set of working operating units has a subset that represents a solution structure.
- Our goal is to produce the logic functions
- $\Psi(x_1, x_2, ..., x_n)$ that define the „ system is working” system event.

- A $\Phi(X)$ logical function can be given, for which
- $\Phi(x_1, x_2, ..., x_n) = 1$ if and only if the operating units given by $(x_1, x_2, \ldots, xn)$ represent a solution structure.

The number of solution structures

- Let us consider the system event where the operating part is the solution structure

$$A = \{(x_1, x_2, ..., x_n) \in \Omega : \Phi(x_1, x_2, ..., x_n) = 1\}$$

- Let S be the characteristic function of the A system event

$$S(x_1, x_2, ..., x_n)$$

from this

$$S(x) = S(x, x, ..., x)$$

- Then the number of solution structures is:

$$2^n S(\frac{1}{2})$$

The number of solution structers: An example

- There are N parallel operating unit

$$\Psi(X) = \Phi(x_1, x_2, ..., x_n) = x_1 \vee x_2 \vee ... \vee x_n$$

- With disjoint clauses (Only n number of clauses!)

$$\Psi(X) = x_1 \vee \overline{x_1} x_2 \vee ... \vee \overline{x_1}\,\overline{x_2}...\overline{x_{n-1}} x_n$$

$$S(x) = x + (1-x)x + ... + (1-x)^{n-1} x$$

$$S(\frac{1}{2}) = \frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^n}$$

$$2^n S(\frac{1}{2}) = 2^n - 1$$

Reliability of process networks

- Step I.:

$$\Phi(x_1, x_2, ..., x_n) = \Phi_0 \wedge \Phi_1 \wedge \Phi_2$$

$$\Phi_0(x_1, x_2, ..., x_n) = \bigwedge_{X \in P} \bigvee_{\substack{j \in J \\ X \in \beta_j}} x_j$$

$$\Phi_1(x_1, x_2, ..., x_n) = \bigwedge_{\substack{j \in J \\ X \in \alpha_j / R}} (\overline{x_j} \vee \bigvee_{\substack{h \in J \\ X \in \beta_h}} x_h)$$

$$\Phi_2(x_1, x_2, ..., x_n) = \bigwedge_{\substack{j \in J \\ P \cap \beta_j = \varnothing}} (\overline{x_j} \vee \bigvee_{\substack{h \in J \\ \beta_j \cap \alpha_h}} x_h)$$

- Step II.:
  - Disjoint normal form
  - With the help of syntactic rules

Reliability of process networks

- Step III.:
  - Calculate $\Psi(x_1, x_2, ..., x_n)$
  - Delete the negative variables from $\Phi(x_1, x_2, ..., x_n)$
- Step IV.:
  - Generate the disjoint closes

$$\Psi(X) = L_1 \vee L_2 \vee ... \vee L_s$$

$$\Psi(X) = L_1 \vee \overline{L_1} L_2 \vee \overline{L_1}\,\overline{L_2} L_3 \vee ... \vee \overline{L_1}\,\overline{L_2} ... \overline{L_{s-1}} L_s$$

  - Remark: This can explode!
  - But it is not necessary to geneerate them explicitly!

Another way: Prime structures

- Step I.:
  - Calculate $\Psi(x_1, x_2, ..., x_n)$

$$\Psi(X) = K_1 \vee K_2 \vee ... \vee K_s$$

$$K_i = x_{i_1} \wedge x_{i_2} \wedge ... \wedge x_{i_l} \qquad i \in \{1, 2, .., s\}$$

  - where $i_1, i_2, ..i_l$ are operating units of the i-th prime structure
- Step II.:
  - Generate the disjoint clauses

$$\Psi(X) = K_1 \vee \overline{K_1} K_2 \vee \overline{K_1}\,\overline{K_2} K_3 \vee ... \vee \overline{K_1}\,\overline{K_2} ... \overline{K_{s-1}} K_s$$

Calculating reliability

- In cases before the disjunction of disjoint clauses

$$\Psi(X) = K_1 \vee \overline{K_1} K_2 \vee \overline{K_1}\,\overline{K_2} K_3 \vee \ldots \vee \overline{K_1}\,\overline{K_2} \ldots \overline{K_{s-1}} K_s$$

$$\Psi(X) = \bigvee_{i=1}^{r} \left(\bigwedge_{j=1}^{l} z_{i_j} \right) \quad ahol \quad z_i = x_i \;\; vagy \;\; \overline{x}_i$$

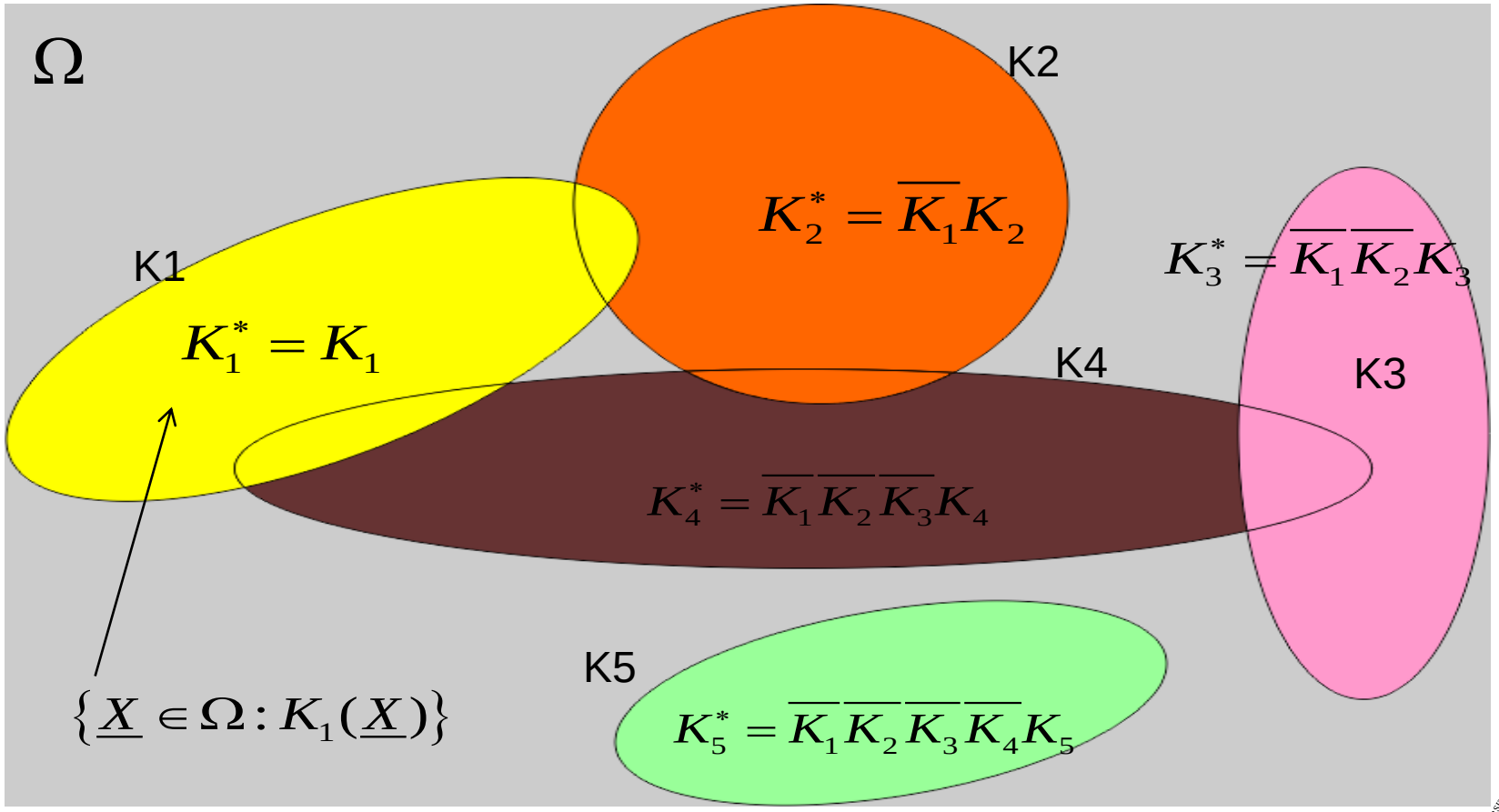
$$u_{i_j} = \begin{cases} 1 & \text{if} \quad z_{i_j} = x_{i_j} \\ 0 & \text{if} \quad z_{i_j} = \overline{x}_{i_j} \end{cases}$$

$$R = P(\Psi(X) = 1) = \sum_{i=1}^{r} \prod_{j=1}^{l} p_{i_j}^{u_{i_j}} (1 - p_{i_j})^{(1-u_{i_j})}$$

Calculating reliability

- It is not necessary to produce the $\Psi(X)$ function explicitly

$$\Psi(X) = K_1 \vee \overline{K_1} K_2 \vee \ldots \vee \overline{K_1}\,\overline{K_2} \ldots \overline{K_{s-1}} K_s$$



Calculating reliability

- This procedure gives the system reliability.

$$K_1^* = K_1$$

$$K_2^* = K_2 \setminus (K_1 \cap K_2)$$

$$K_3^* = K_3 \setminus ((K_1 \cap K_3) \setminus ((K_2 \cap K_3) \setminus (K_1 \cap K_2 \cap K_3)))$$

$$P(K_3^*) = P(K_3) - (P(K_1K_3) - (P(K_2K_3) - P(K_1K_2K_3)))$$

$$procedure(\ inpSTR\ )$$

$$while(\ K_i = nextPRIM\ != \varnothing\ )$$

$$\left\{ \begin{array}{l} K_i^* = K_i \setminus (\bigcup_{j=1}^{i-1} K_j^* \cap K_i) \\ R = R + P(K_i^*) \end{array} \right\}$$

A third way: Cutting sets

- A set of operating units H is called to be a cutting set if in case the operating units are not working, then the system is also not working.
- Let E be the set of cutting sets, then:

$$R = 1 - \sum_{H \in E} \prod_{i \in H} [1 - p_i] \prod_{i \notin H} [p_i]$$

- The aggregate value is the sum of the probabilities of events when the system is not working.

$$E = \{(x_1, x_2, ..., x_n) \in \Omega : \Psi(x_1, x_2, ..., x_n) = 0\}$$

$$R = P(E)$$

Example: Function of solution structures

- Step I.: $\Phi(x_1, x_2, x_3) = \Phi_0 \wedge \Phi_1 \wedge \Phi_2$

$$\Phi_0(x_1, x_2, x_3) = x_3$$
$$\Phi_1(x_1, x_2, x_3) = (\overline{x_3} \vee x_1 \vee x_2) \wedge (\overline{x_3} \vee x_2)$$
$$\Phi_2(x_1, x_2, x_3) = (\overline{x_1} \vee x_3) \wedge (\overline{x_2} \vee x_3)$$

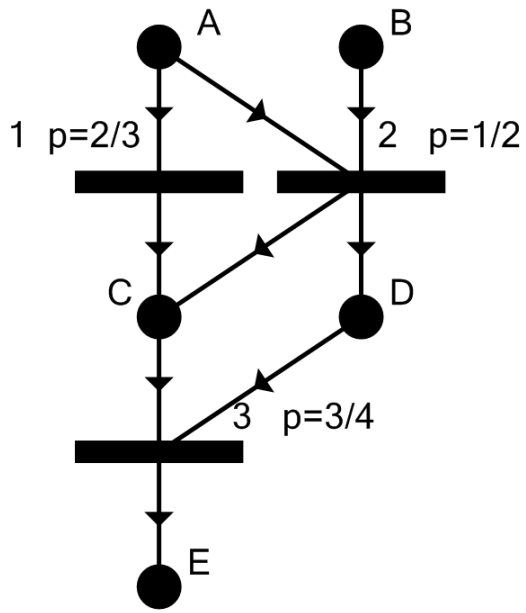
- Step II.:

$$\Phi(x_1, x_2, x_3) = x_2 \wedge x_3$$

- Step III.: None

$$\Psi(x_1, x_2, x_3) = x_2 \wedge x_3$$
$$Q(x_1, x_2, x_3) = x_2 * x_3$$

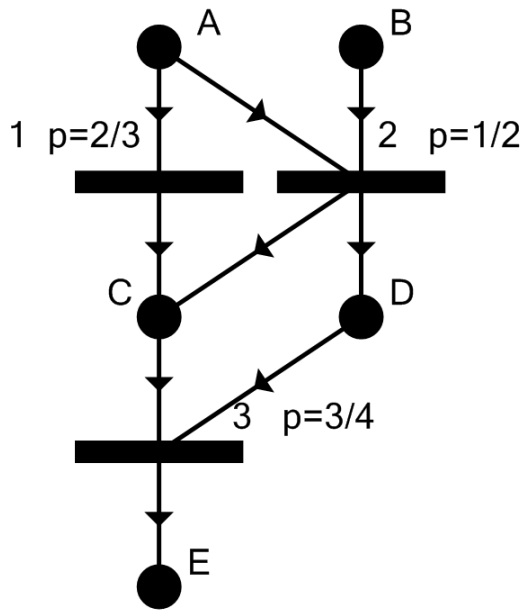
- Step IV.: None



$$R = \frac{1}{2} * \frac{3}{4} = \frac{3}{8}$$

Example: Prime structures

- There is only one prime structure {2, 3}



- Step I.:
  - There is only one clause

$$\Psi(x_1, x_2, x_3) = x_2 \wedge x_3$$

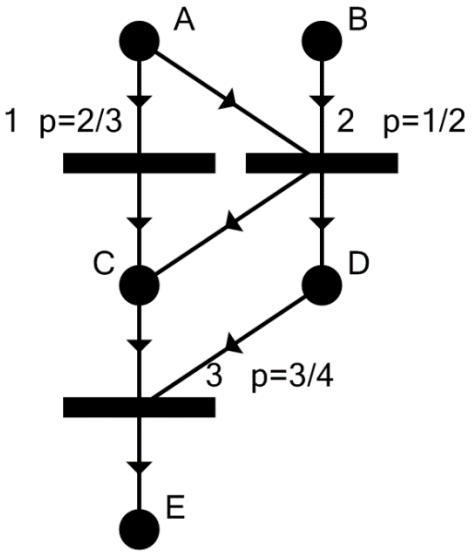
$$Q(x_1, x_2, x_3) = x_2 * x_3$$

- Step II.:
  - None

$$R = \frac{1}{2} * \frac{3}{4} = \frac{3}{8}$$

Example: Cutting sets

- A set of operating units is a cutting set, if they are switched off, the system does no work.
- Let E be the set of cutting sets
- E={ {2}, {3}, {1,3}, {2,3}, {1,2}, {1,2,3} }



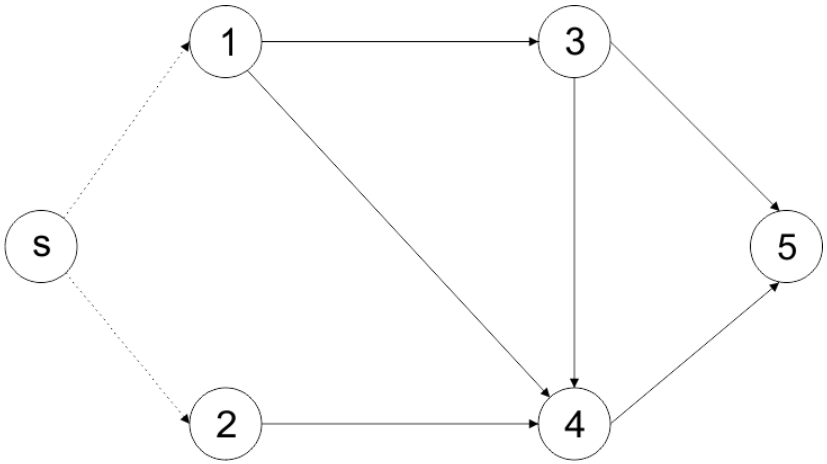
$$R = 1 - \sum_{H \in E} \prod_{i \in H} [1 - p_i] \prod_{i \notin H} [p_i]$$

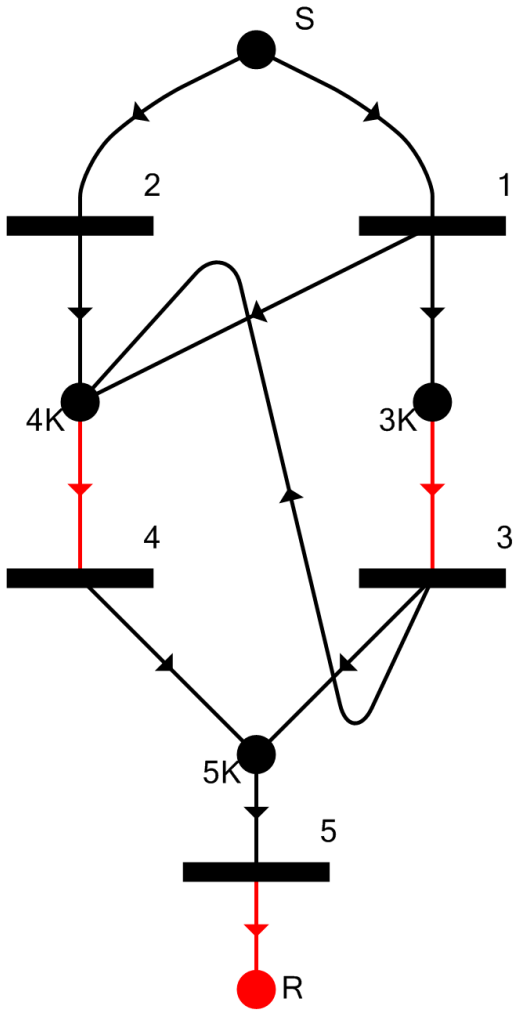
$$R = \frac{3}{8}$$

R = 1 – ((2/3)*(1/2)*(3/4) + (2/3)*(1/2)*(1/4) + …) = 18/48

Conventional model and P-graph model

Example1





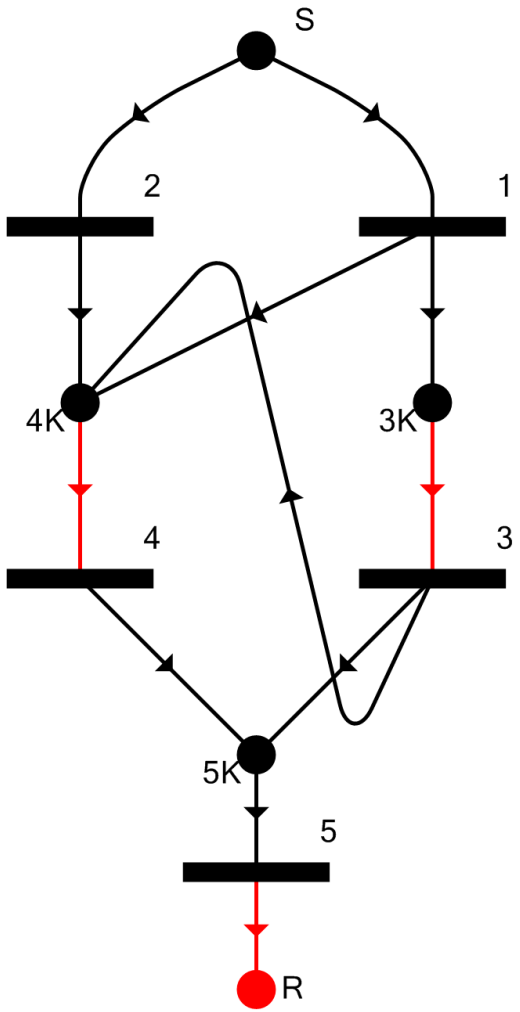
Based on solution structures, Example 1

Step I.: $\Phi(x_1, x_2, ..., x_5) = \Phi_0 \wedge \Phi_1 \wedge \Phi_2$

$$\Phi_0(X) = x_5$$

$$\Phi_1(X) = (\overline{x_5} \vee x_3 \vee x_4) \wedge (\overline{x_4} \vee x_1 \vee x_2 \vee x_3) \wedge (\overline{x_3} \vee x_1)$$

$$\Phi_2(X) = (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee x_4) \wedge (\overline{x_3} \vee x_5) \wedge (\overline{x_4} \vee x_5)$$



Step II. III. and IV.:

$$R = 0,8748$$

Based on prime structures, Example 1

- Step I.:
- {1,3,5}, {1,4,5}, {2,4,5,}

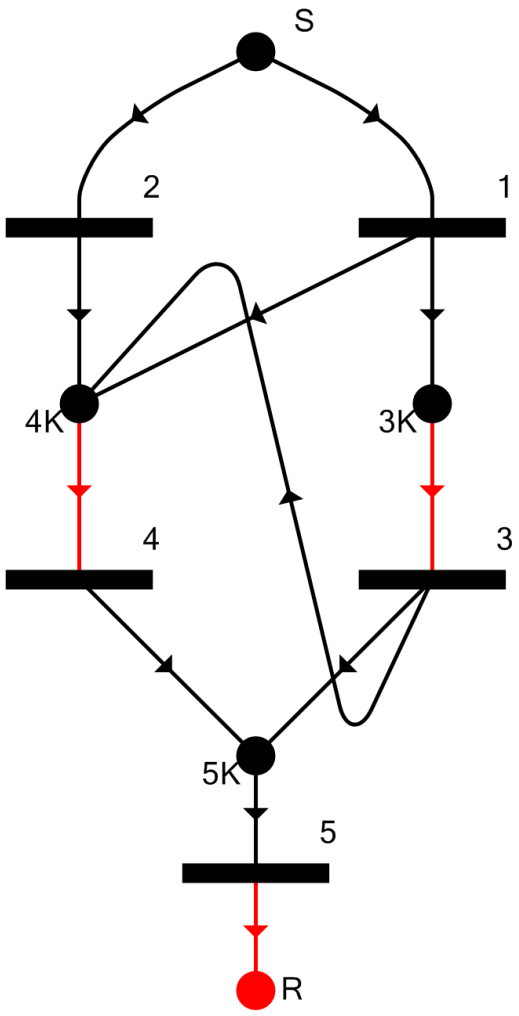
$$\Psi(X) = (x_1 \wedge x_3 \wedge x_5) \vee (x_1 \wedge x_4 \wedge x_5) \vee (x_2 \wedge x_4 \wedge x_5)$$

- Step II.:

$$\Psi(X) = (x_1 \wedge x_3 \wedge x_5) \vee (x_1 \wedge \bar{x}_3 \wedge x_4 \wedge x_5) \vee (\bar{x}_1 \wedge x_2 \wedge x_4 \wedge x_5)$$

$$Q(X) = x_1 * x_3 * x_5 + x_1 * (1 - x_3) * x_4 * x_5 + (1 - x_1) * x_2 * x_4 * x_5$$

$$R = 0,8748$$



Based on prime structures, Example 1

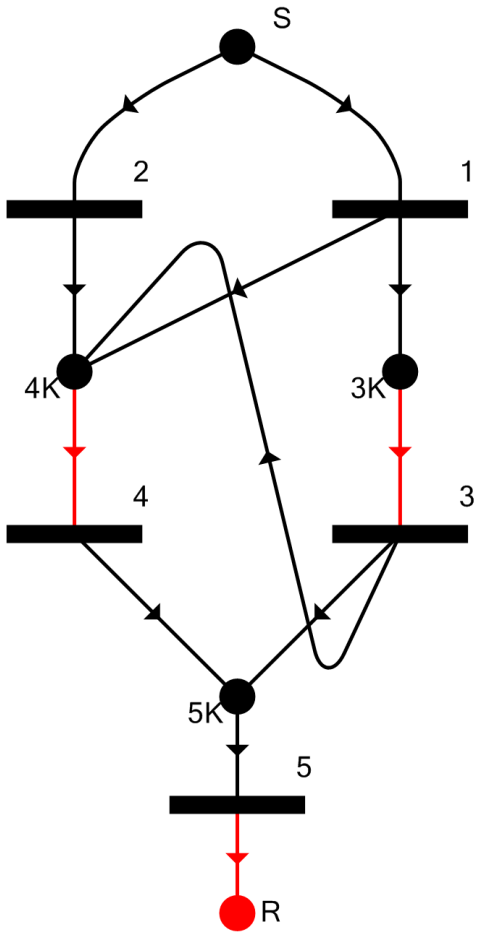
- The steps of the algorithm

$$K_1^* \rightarrow p_1 = P(K_1)$$
$$K_1^* \rightarrow p_1 = P((x_1 x_3 x_5)) = 0,729$$
$$K_2^* \rightarrow p_2 = P(K_2) - P(K_1 K_2)$$
$$K_2^* \rightarrow p_2 = P((x_1 x_4 x_5)) - P((x_1 x_3 x_4 x_5))$$
$$= 0,729 - 0,6561$$
$$K_3^* \rightarrow p_3 = P(K_3) - P(K_1 K_3) - (P(K_2 K_3) - P(K_1 K_2 K_3))$$
$$K_3^* \rightarrow p_3 = P(x_2 x_4 x_5) - P(x_1 x_2 x_3 x_4 x_5) - ((P(x_1 x_2 x_4 x_5) - P(x_1 x_2 x_3 x_4 x_5))$$
$$= 0,729 - 0,6561$$

$$R = 0,8748$$

Based on cutting sets, Example1

- It is very exhausting to give explicitely
- {5},{1,2},{1,4},{1,5},{2,5},{3,5}, ,{4,5},{1,2,3}, …



Gas network

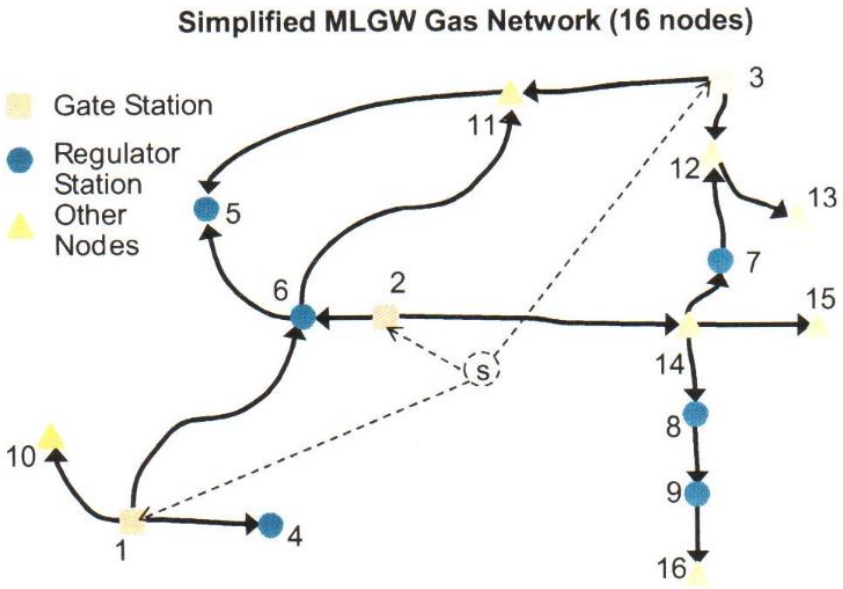
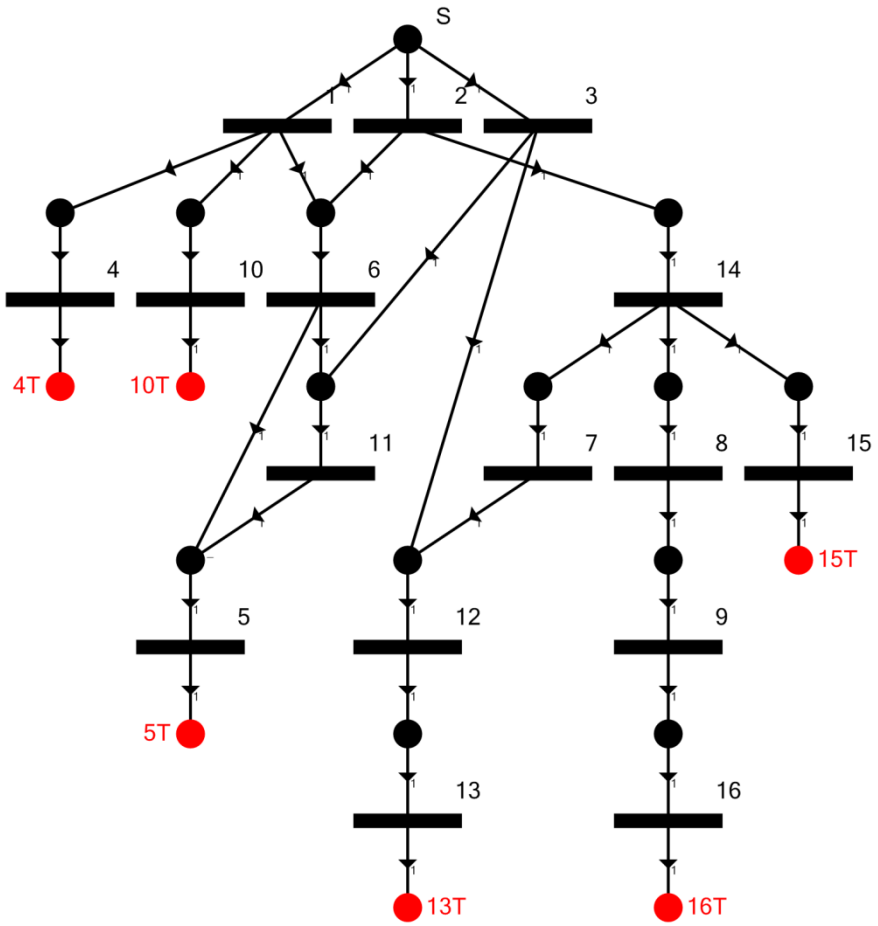


Fig. 5. A natural gas transmission network in Shelby County, TN [19].



Reliability and synthesis

- A maximal structure is given.
- Which is the most reliable solution?
- Naturally the maximal structure!
- What is value of reliability of a solution?

$$X \in \Omega \text{ és } \Phi(X) = 1$$

$$B_X = \{Y \in \Omega : Y \leq X\}$$

$$A_X = \{Y \in \Omega : \Psi(Y) = 1 \text{ és } Y \in B_X\}$$

$$R_X = P(A_X \mid B_X) = P(A_X) / P(B_X)$$

$$R_{(1,1,\ldots,1)} = P(\Psi(X) = 1) / P(\Omega) = P(\Psi(X) = 1) = R$$

Reliability and synthesis, Case I.

- How reliable units to be used?
- For a given system, we are looking for the solution for which in the dependence of the reliability of the operating units costs minimum and it suffices the confidence value for the operability of the system.
- Let be

$$Y = (y_1, y_2, \ldots, y_n) \quad ahol \quad y_i = P(x_i = 1)$$

- The mathematical model:

$$Q(Y) \geq R_{küszöb}$$

$$Y \in [0,1]^n$$

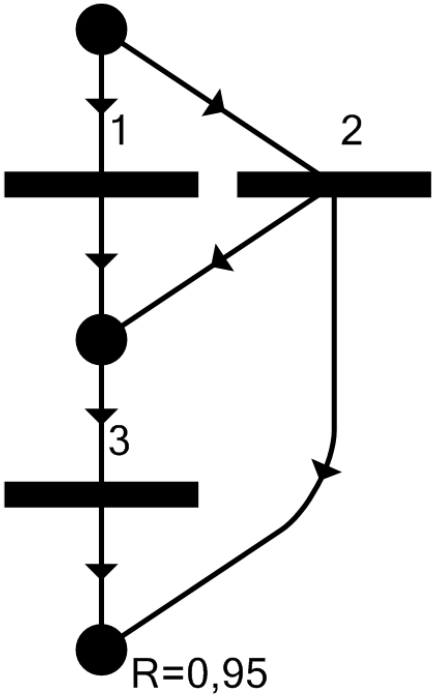
$$\overline{C(Y) \rightarrow \min}$$

Reliability and synthesis, Case I. (2)

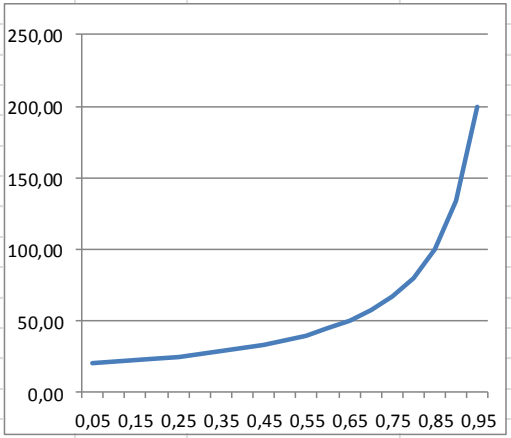
$$y_2 + y_1(1-y_2)y_3 \geq 0,95$$

$$y_1, y_2, y_3 \in [0,1]$$

$$10/(1.005-y_1)+50/(1.005-y_2)+10/(1.005-y_3) \rightarrow \min$$



Eset_I	Gép1	Gép2	Gép3	Rendszer	Küszöb
	X1	X2	X3	Q=x2+x1(1-x2)x3	
Megbízhatóság	0,9343698	0,6030567	0,9355258	0,950035567	0,95
KöltségParaméter	10	50	10		
Költség_FGV	86,482579	111,87101	87,355934	285,7095185	Költség
0,05	20,00				
0,1	21,05				
0,15	22,22				
0,2	23,53				
0,25	25,00				
0,3	26,67				
0,35	28,57				
0,4	30,77				
0,45	33,33				
0,5	36,36				
0,55	40,00				
0,6	44,44				
0,65	50,00				
0,7	57,14				
0,75	66,67				
0,8	80,00				
0,85	100,00				
0,9	133,33				
0,95	200,00				
1	400,00				



Reliability and synthesis, Case II.

- Which is the most critical event?
- Which is the most probable critical event, for which in case of its supervention the system is not working?

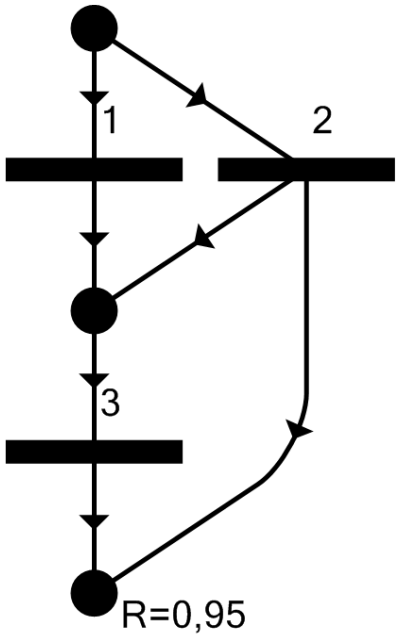
$$p_{krit} = \max_{\{X \in \Omega \text{ és } \Psi(X)=0\}} P(A_X), \quad A_X = \{Y : Y \in \Omega, Y \leq X\}$$

$$Q(X) = 0$$
$$X \in \{0,1\}^n$$
$$\sum_{i=1}^{n} p_i^{x_i} (1-p_i)^{(1-x_i)} \rightarrow \min = p_{krit}$$

- It is proposed to investigate the conditions for the joint failure of the operating units belonging to the zero elements of X!

Reliability and synthesis, Case II. (2)

- The 2nd and 3rd operating units' combined shut down is critical.



	Gép1	Gép2	Gép3	Rendszer	Küszöb
	X1	X2	X3	Q=x2+x1(1-x2)x3	
Megbízhatóság	0,927696	0,642646	0,927215	0,950032501	0,95
KöltségParaméte	10	100	10		
Költség_FGV	136,4183	279,0535	135,529	551,0008238	KTSG
RendszerLeállás	1	0	0	0	0
	1	0,357354	0,072785	0,026010007	

Reliability and synthesis, Case III.

- Which operating units should be replicated and how many times?
- A network is given with the reliability of its operating units, costs and a given reliability threshold value for the system.
- We are looking for a system produced by the multiplication of the operating units, that suffices the given reliability threshold and has minimal cost.

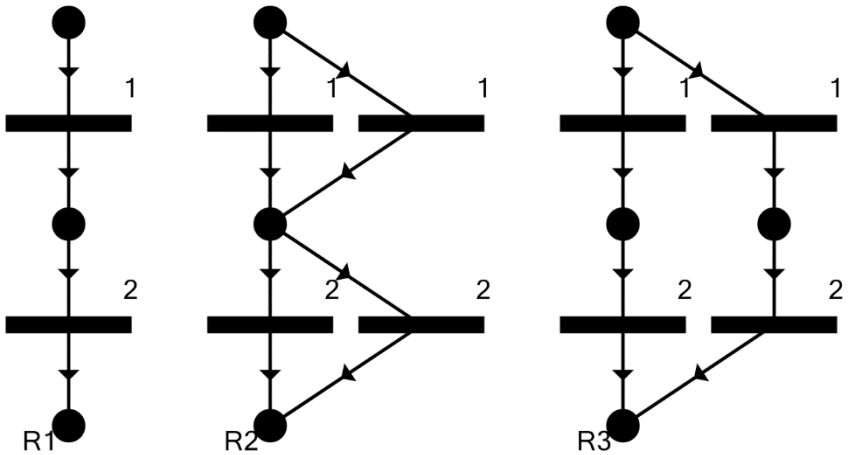
Reliability and synthesis, Case III. (2)

- Should we multiply the units by themselves one by one, or fittingly chosen subsystems?

$$R_1 = p_1 * p_2$$

$$R_2 = (1-(1-p_1)*(1-p_1))*(1-(1-p_2)*(1-p_2))$$

$$R_3 = (1-(1-p_1p_2)*(1-p_1p_2))$$

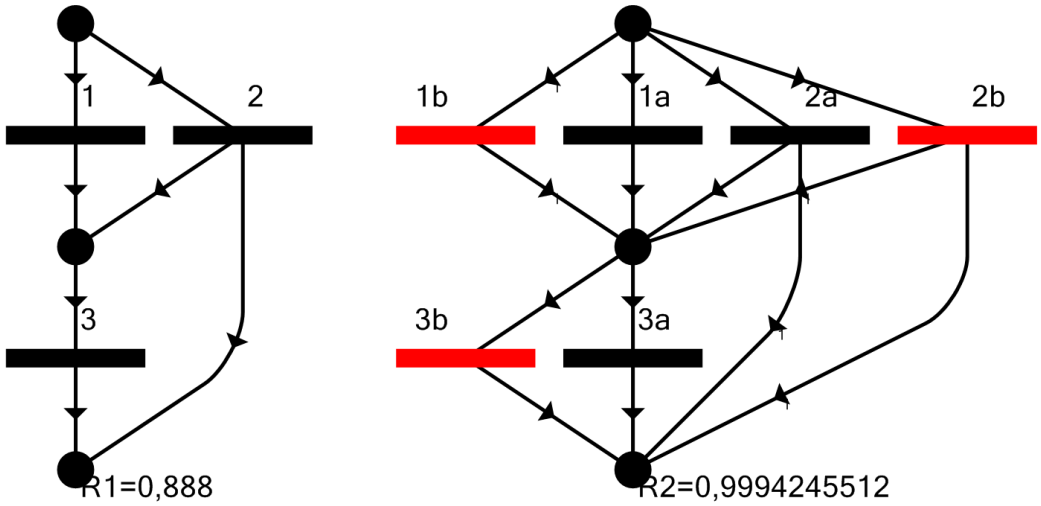


$$R_2 \geq R_3$$

- This holds in general!

Reliability and synthesis, Case III. (3)

- According to this, it is sufficient to multiply the operating units by themselves, until the given confidence level is reached.



Reliability and synthesis, Case III. (4)

- If we multiply the operating units, we have to produce the characteristic polynom of the new system.

$$\Psi(x_1, x_2, ..., x_n) \quad \rightarrow \quad Q(x_1, x_2, ..., x_n)$$

$$R(k_1, k_2, ..., k_n, x_1, x_2, ..., x_n) = Q(1-(1-x_1)^{k_1}, 1-(1-x_2)^{k_2}, ..., 1-(1-x_n)^{k_n})$$

- The mathematical model of the problem:

$$k_1, k_2, ..., k_n \in N$$

$$R(k_1, k_2, ..., k_n, p_1, p_2, ..., p_n) \geq p_{küszöb}$$

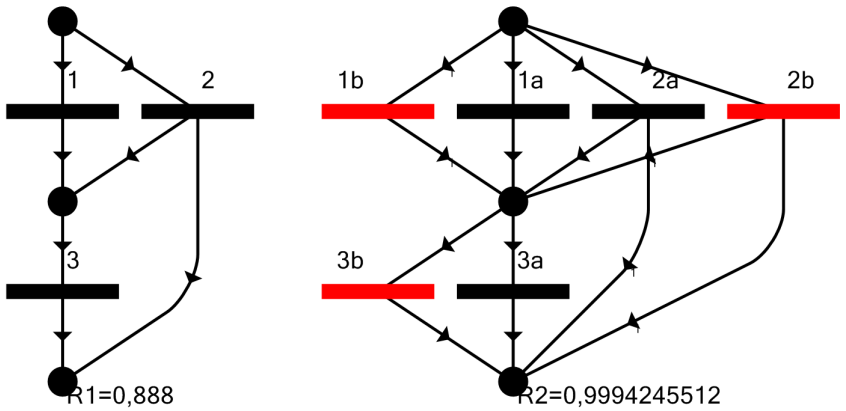
$$C(k_1, k_2, ..., k_n) = k_1 b_1 + k_2 b_2 + ... + k_n b_n \rightarrow \min$$

Solution of a synthesis problem

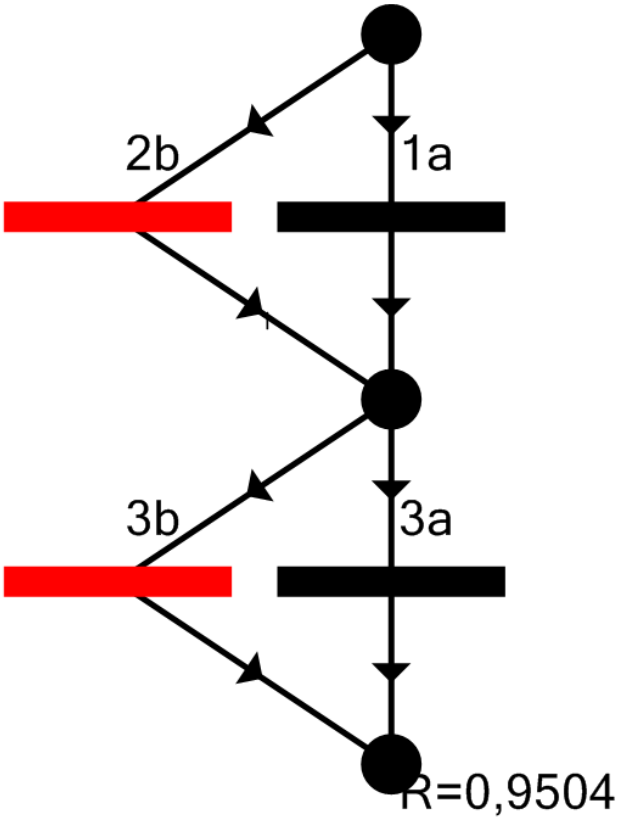
- The problem:

$$p_1 = 0,8 \quad p_2 = 0,6 \quad p_3 = 0,9$$
$$c_1 = 10 \quad c_1 = 50 \quad c_1 = 20 \quad p_{küszöb} = 0,95$$

$$\Psi(x_1, x_2, x_3) = x_2 + x_1 \bar{x}_2 x_3$$
$$Q(x_1, x_2, x_3) = x_2 + x_1(1 - x_2)x_3$$
$$R(2, x_1, x_2, x_3) = Q(2x_1 - x_1^2, 2x_2 - x_2^2, 2x_3 - x_3^2)$$

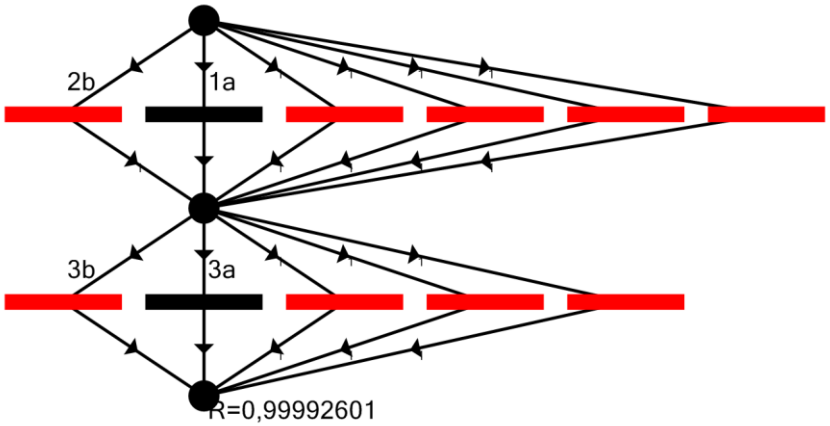


Solution of a synthesis problem, R>=0,95.



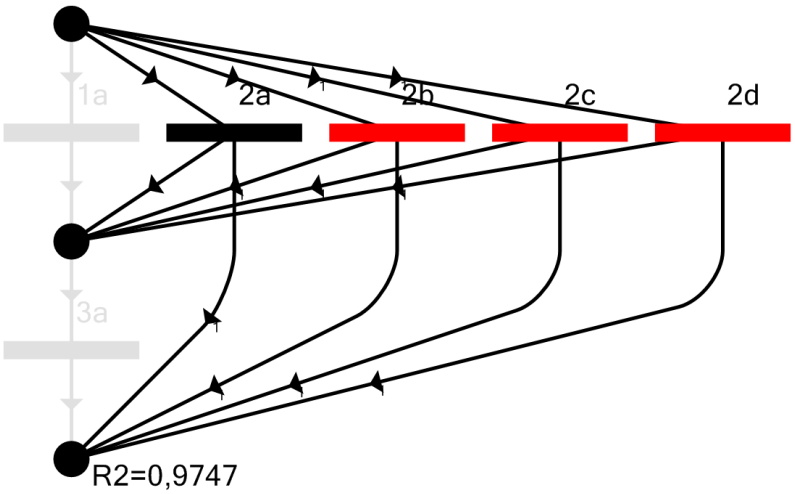
X1	X2	X3	Q=x2+x1(1-x2)x3	Többszörözés
0,8	0,6	0,9	0,888	1X
0,96	0,84	0,99	0,992064	2X
0,992	0,936	0,999	0,999424512	3X
0,9984	0,9744	0,9999	0,999956484	4X
0,99968	0,98976	0,99999	0,999996621	5X
0,999936	0,995904	0,999999	0,999999734	6X
0,9999872	0,9983616	0,9999999	0,999999979	7X
0,99999744	0,99934464	0,99999999	0,999999998	8X
0,96	0	0,99	**0,9504**	**0,95**
0	0	0	1	1X
1	0	1	2	2X
0	0	0	3	3X
0	0	0	4	4X
0	0	0	5	5X
0	0	0	6	6X
0	0	0	7	7X
0	0	0	8	8X
2	0	2		Db
10	50	20	**60**	Beruházás
1	0	1	1	Működőképes

Solution of a synthesis problem, R>=0,9999



X1	X2	X3	Q=x2+x1(1-x2)x3	Többszörözés
0,8	0,6	0,9	0,888	1X
0,96	0,84	0,99	0,992064	2X
0,992	0,936	0,999	0,999424512	3X
0,9984	0,9744	0,9999	0,999956484	4X
0,99968	0,98976	0,99999	0,999996621	5X
0,999936	0,995904	0,999999	0,999999734	6X
0,9999872	0,9983616	0,9999999	0,999999979	7X
0,99999744	0,99934464	0,99999999	0,999999998	8X
0,999936	0	0,99999	**0,999926001**	**0,9999**
0	0	0	1	1X
0	0	0	2	2X
0	0	0	3	3X
0	0	0	4	4X
0	0	1	5	5X
1	0	0	6	6X
0	0	0	7	7X
0	0	0	8	8X
6	0	5		Db
10	50	20	**160**	Beruházás
1	0	1	1	Működőképes

Solution with other parmeters.



X1	X2	X3	Q=x2+x1(1-x2)x3	Többszörözés
0,7	0,6	0,8	0,824	1X
0,91	0,84	0,96	0,979776	2X
0,973	0,936	0,992	0,997773824	3X
0,9919	0,9744	0,9984	0,999752012	4X
0,99757	0,98976	0,99968	0,999971848	5X
0,999271	0,995904	0,999936	0,999996752	6X
0,9997813	0,9983616	0,9999872	0,999999621	7X
0,99993439	0,99934464	0,99999744	0,999999955	8X
0	0,9744	0	0,9744	0,95
0	0	0	1	1X
0	0	0	2	2X
0	0	0	3	3X
0	1	0	4	4X
0	0	0	5	5X
0	0	0	6	6X
0	0	0	7	7X
0	0	0	8	8X
0	4	0		Db
10	10	10	40	Beruházás
0	1	0	1	Feltétel

Solution of the relaxed model

The upper bound of the multiplication has to be determined for each operating unit.
We get an exact upper bound for the costs, but not yet for the multiplication of the units.

$$y_1, y_2, \ldots, y_n \in R_+$$

$$R(y_1, y_2, \ldots, y_n, p_1, p_2, \ldots, p_n) \geq p_{küszöb}$$

$$C(y_1, y_2, \ldots, y_n) = y_1 b_1 + y_2 b_2 + \ldots + y_n b_n \rightarrow \min$$

Y1	Y2	Y3	Q=(y1,y2,y3)	Küszöb
6,312369	0,000427	4,201173	0,999898417	0,9999
0,8	0,7	0,9		
10	50	20	147,1685129	Költség

$$Q(x_1, x_2, x_3) = x_2 + x_1(1 - x_2)x_3$$

$$R(Y, P) = (1 - (1 - p_2)^{y_2}) + (1 - (1 - p_1)^{y_1}) * (1 - p_2)^{y_2} * (1 - (1 - p_3)^{y_3})$$

Exact upper bound for multiplication

- With the help of prime structures.

$$\Psi(X) = K_1 \vee K_2 \vee \ldots \vee K_s$$

$$K_i = x_{i_1} \wedge x_{i_2} \wedge \ldots \wedge x_{i_l} \qquad i \in \{1, 2, .., s\}$$

$$a_j = \min_{z \in N} \left\{ z : (1-(1-p_{j_1})^z) * (1-(1-p_{j_2})^z) * \ldots * (1-(1-p_{j_l})^z) \geq p_{küszöb} \right\}$$

$$k_i = \max_{j=(1,2,\ldots,s)} \left\{ v_j = \begin{cases} a_j & if \quad i \in \{j_1, j_2, \ldots, j_l\} \\ 0 & otherwise \end{cases} \right\}$$

Boundaries

X1	X2	X3	Q=x2+x1(1-x2)x3	Többszörözés	{1,3}	{2}
0,8	0,7	0,9	0,916	**1X**	0,72	0,7
0,96	0,91	0,99	0,995536	**2X**	0,9504	0,91
0,992	0,973	0,999	0,999757216	**3X**	0,991008	0,973
0,9984	0,9919	0,9999	0,999986231	**4X**	0,99830016	0,9919
0,99968	0,99757	0,99999	0,999999198	**5X**	0,999670003	0,99757
0,999936	0,999271	0,999999	0,999999953	**6X**	**0,999935**	0,999271
0,9999872	0,9997813	0,9999999	0,999999997	**7X**	0,9999871	0,9997813
0,99999744	0,99993439	0,99999999	1	**8X**	0,99999743	**0,99993439**
0,999936	0	0,99999	**0,999926001**	**0,9999**		
0	0	0	1	**1X**		
0	0	0	2	**2X**		
0	0	0	3	**3X**		
0	0	0	4	**4X**		
0	0	1	5	**5X**		
1	0	0	6	**6X**		
0	0	0	7	**7X**		
0	0	0	8	**8X**		
6	0	5		**Db**		
50	80	50	**550**	**Beruházás**		
1	0	1	1	**Működőképes**		

Reliability and synthesis. Calculations.

- Formulas:

$$\Psi(x_1, x_2, \ldots, x_n) \quad \rightarrow \quad Q(x_1, x_2, \ldots, x_n)$$

$$R(k_1, k_2, \ldots, k_n, x_1, x_2, \ldots, x_n) = Q(1-(1-x_1)^{k_1}, 1-(1-x_2)^{k_2}, \ldots, 1-(1-x_n)^{k_n})$$

$$R(k, x_1, x_2, \ldots, x_n) = Q(1-(1-x_1)^{k}, 1-(1-x_2)^{k}, \ldots, 1-(1-x_n)^{k})$$

$$R(2k, x_1, x_2, \ldots, x_n) = R(k, 2x_1 - x_1^2, 2x_2 - x_2^2, \ldots, 2x_n - x_n^2)$$

- Iterations:

$$R(k_1, k_2, \ldots, k_n, p_1, p_2, \ldots, p_n) =$$

$$Q(1-(1-p_1)^{k_1}, 1-(1-p_2)^{k_2}, \ldots, 1-(1-p_n)^{k_n})$$

$$\forall i \quad u_{k_i} = 1-(1-p_1)^{k_i}$$

$$R(k_1+1, k_2+1, \ldots, k_n+1, p_1, p_2, \ldots, p_n) =$$

$$Q(p_1 + u_{k_1}(1-p_1), p_2 + u_{k_2}(1-p_2), \ldots, p_n + u_{k_n}(1-p_n))$$

Summary

- The reliability of a process systems was defined.
- The concept of characteristic polynoms was introduced.
- A closed formula for the number of solutions was given.
- Three methods for generating reliability and a way for calculating it were introduced.
- A transposition rule to P-graph model was given.
- Case studies were investigated.
- Three synthesis problem, that are based on the measure of reliability were raised and solved.
- Further problems are to be solved.

Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: axioms and theorems. Chem. Eng. Sci. 1992, 47, 1973.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Combinatorial algorithms for process synthesis. Comput. Chem. Eng. 1992, 16, S313.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- http://www.p-graph.com

References

- J. von Neumann "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components" Automata Studies, C. E: Shannon and J. McCarthy, eds. Princton University Press 1956., pp. 43-98
- Fulop, J., B. Imreh, and F. Friedler, On the Reformulation of Some Classes of PNS-Problems as Set Covering Problems, Acta Cybernetica, 13, 329-337 (1998).

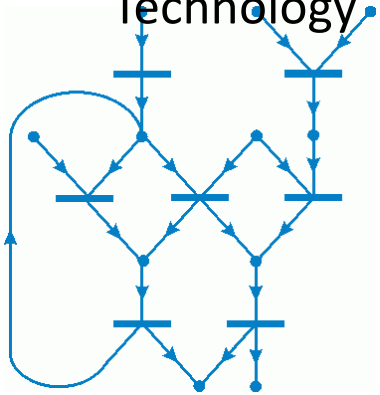
Sustainable Energy Supply Chain Synthesis Using the P-graph Methodology

Outline

- Sustainability metrics (ecological footprint, emergy)
- Supply chain design by P-graph framework
- Incorporation of sustainability constraints
- Illustrative example

Cooperation

University of Pannonia, Department Computer Science and Systems Technology



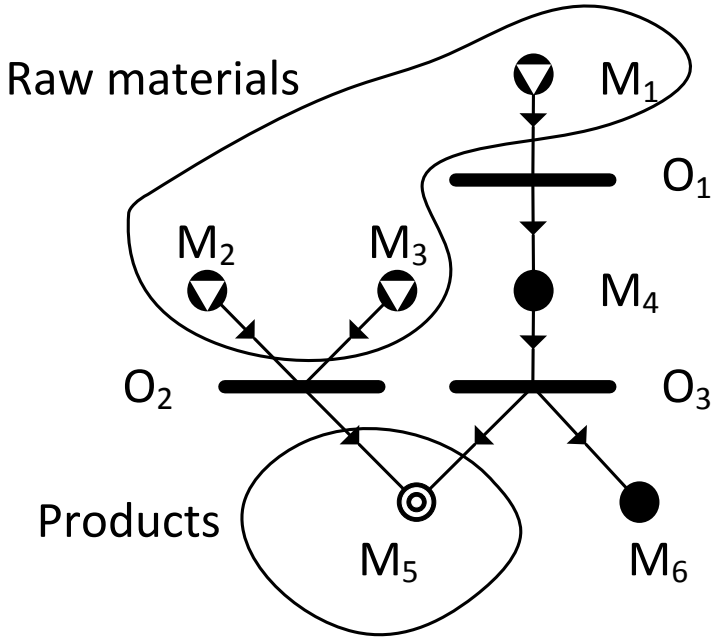
U.S. EPA, Office of Research and Development

Ecological footprint

Ecological footprint

Objective

P-graph methodology



Sustainability

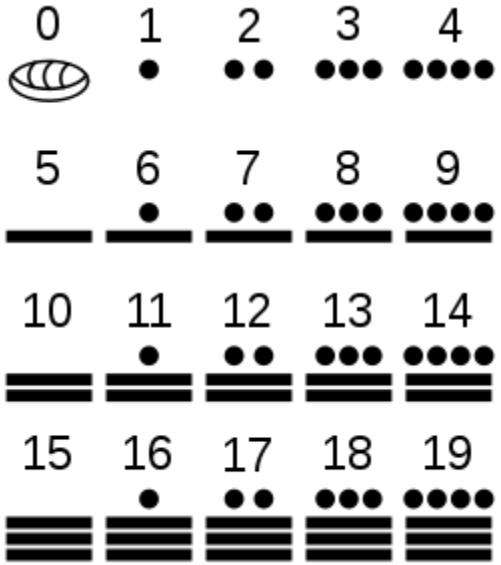
Environmental protection

Environmental protection

- Environmental protection is an older concept than sustainability but the two are closely related
- **Environmental protection**: do not damage the environment unnecessarily, protect
  - the water
  - the soil
  - the air
  - the forest
- Otherwise society will pay the price
  - if not now then later

Learn from the mistakes of the past: Maya empire

Art



Science

Maya empire

Pyramids

Huge cities

Maya empire

- Rapid collapse around 900
- Possible reasons:
  - **overpopulation**
  - attack
  - trade collapse
  - **climate change**
  - epidemic
  - **agriculture fails**

Mesopotamia

- Advanced irrigation system
- The city of Mashkan-shapir was suddenly abandoned
  - the irrigation destroyed the soil in the long run by accumulating mineral salts
- In the San Joaquin valley (USA, California) it happens again

Salt-damaged fields in California's San Joaquin Valley

Big civilizations

- Maya
- Aztecs
- Inca
- Egypt
- Roman
- Mongol
- …
- They failed because of
  - attack
  - crumbled under they own weight, **they were not sustainable**

Human beings always disturb the environment

Pollution

It is not dark but smog

London

Monet

Peking

after rain before rain

Pollution

- Great London smog: December 1952 – March 1953
- **Several thousand fatality**
- It was a kind of whistle blow

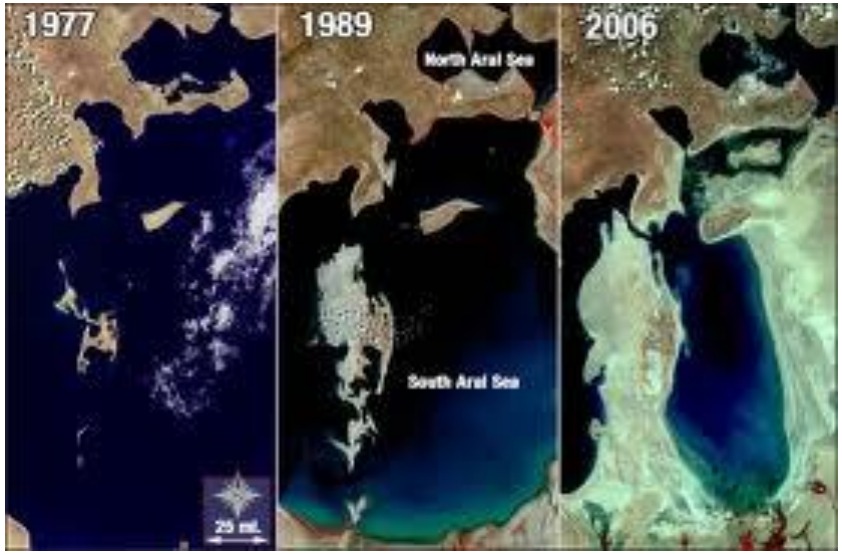
Extreme weather

storms

cold

desertification

Water shortage



Aral Sea

Raw material shortage

- August 2010, Robert Friedland: We need **more copper in the next 20 years than was mined in the last 110 years**, those of us in the business don't have any idea where this metal is going to come from

Biodiversity reduction

- Hunting
- Destroying of habitats

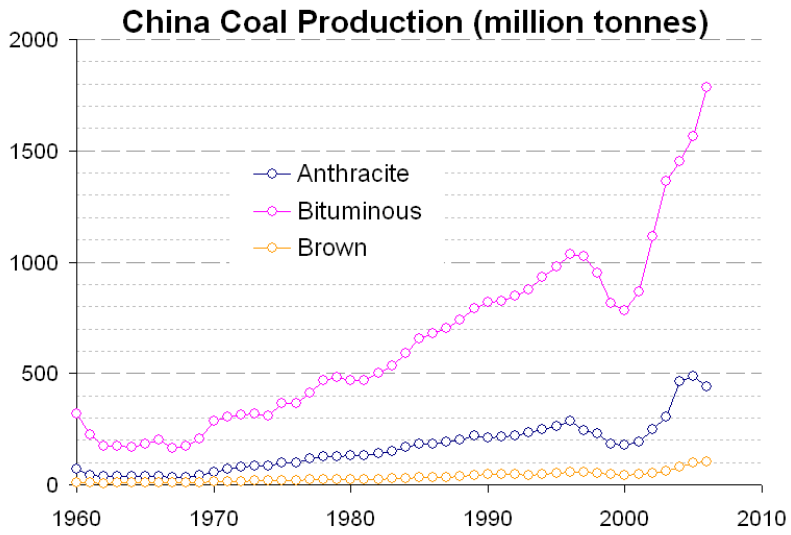
- Extinction of species is a natural phenomenon but **humans increased its rate**

How much water is needed to produce a cup of coffee?

140 L of water (Chapagain and Hoekstra, 2007)

Coal power plans in China

- Total output: $1.95*10^{12}$ kilowatt-hours / year (**275 Paks**)
- $2.38*10^{9}$ t / yr coal
  - more than **USA, EU and Japán together**
  - 13 people dies daily



Coal power plans in China

- In each 7-10 days a new coal power plant is built New York Times, 2006
- India also rely on coal more and more

The concept of sustainability

Definition of sustainability

- United Nations, 1987 Brundtland report:

" Sustainable development is development that meets the needs of the present without compromising the ability of future generations to meet their own needs."

Why is sustainability important?

- United States Census Bureau: human population 2.5 to 6.4 billion 1950-2005
- U.N. Development Program, increase in consumption expenditures 1970-95: Industrial nations 2x and developing nations 2.7X
- Photosynthetic world terrestrial net primary production: 38% used by humans (Running, S.W., 2012. Science, 337, pp. 1458-1459)
- Photosynthetic world primary production: 20% used by humans (Imhoff et al. 2004, Nature, 429)

Motivation

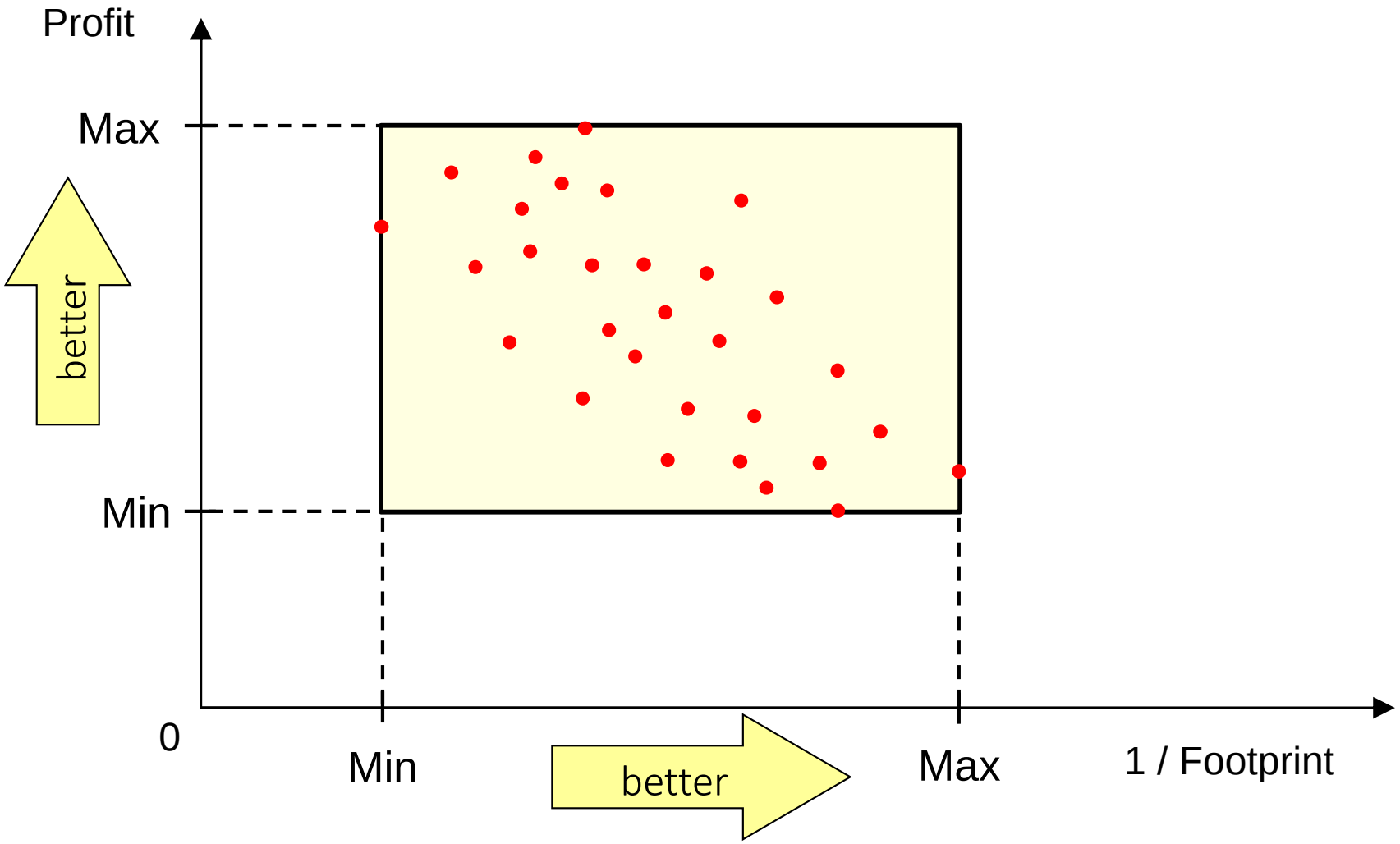
vs.

Sustainability

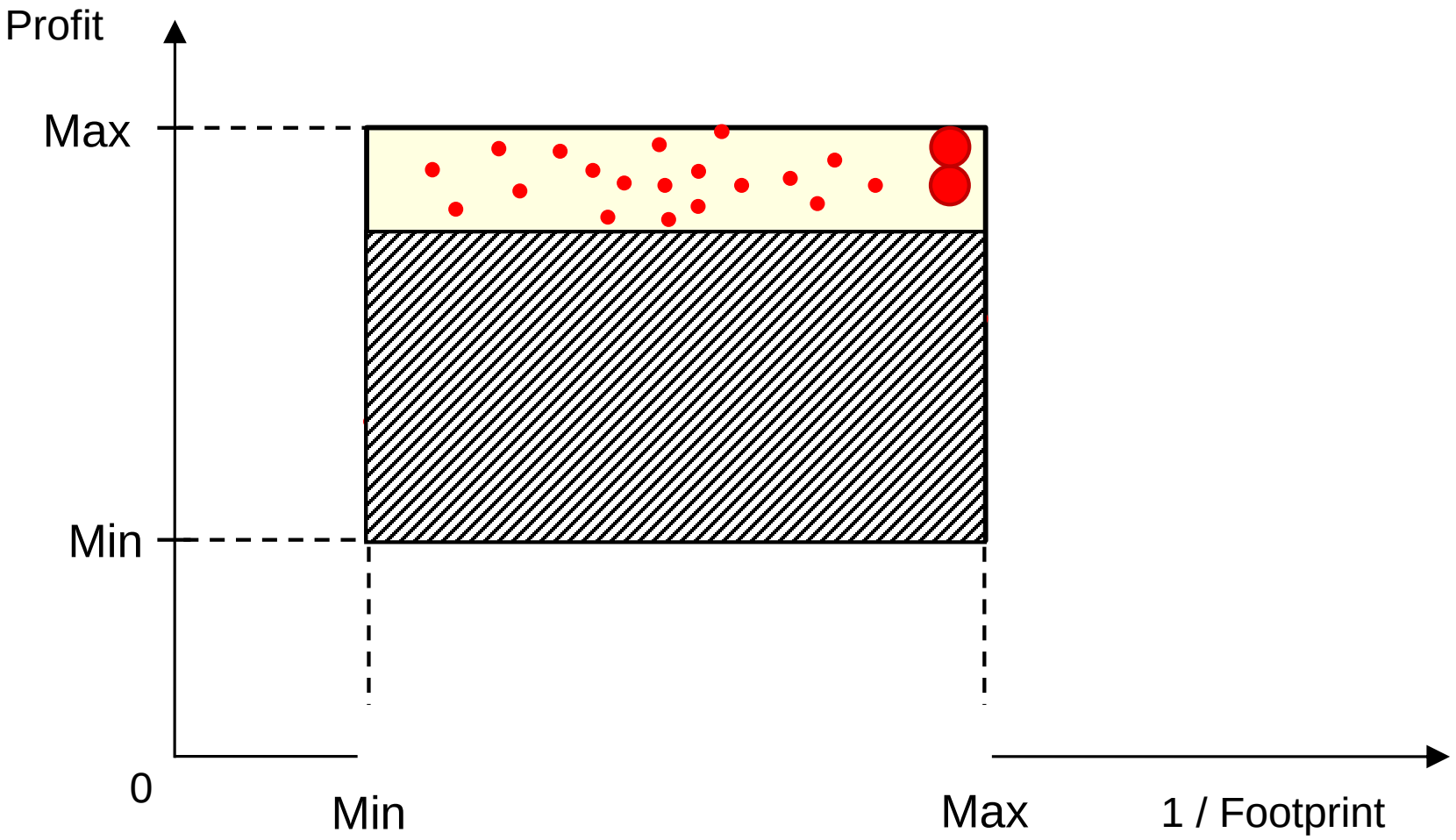
Sustainability vs. Economy?

- Some says that sustainability and economy can not be reconciled
  - developing nations want to grow first and deal with sustainability later
- The two should be done parallel
- Sustainability map: display different solutions in terms of profit and sustainability
  - there is room to improve both

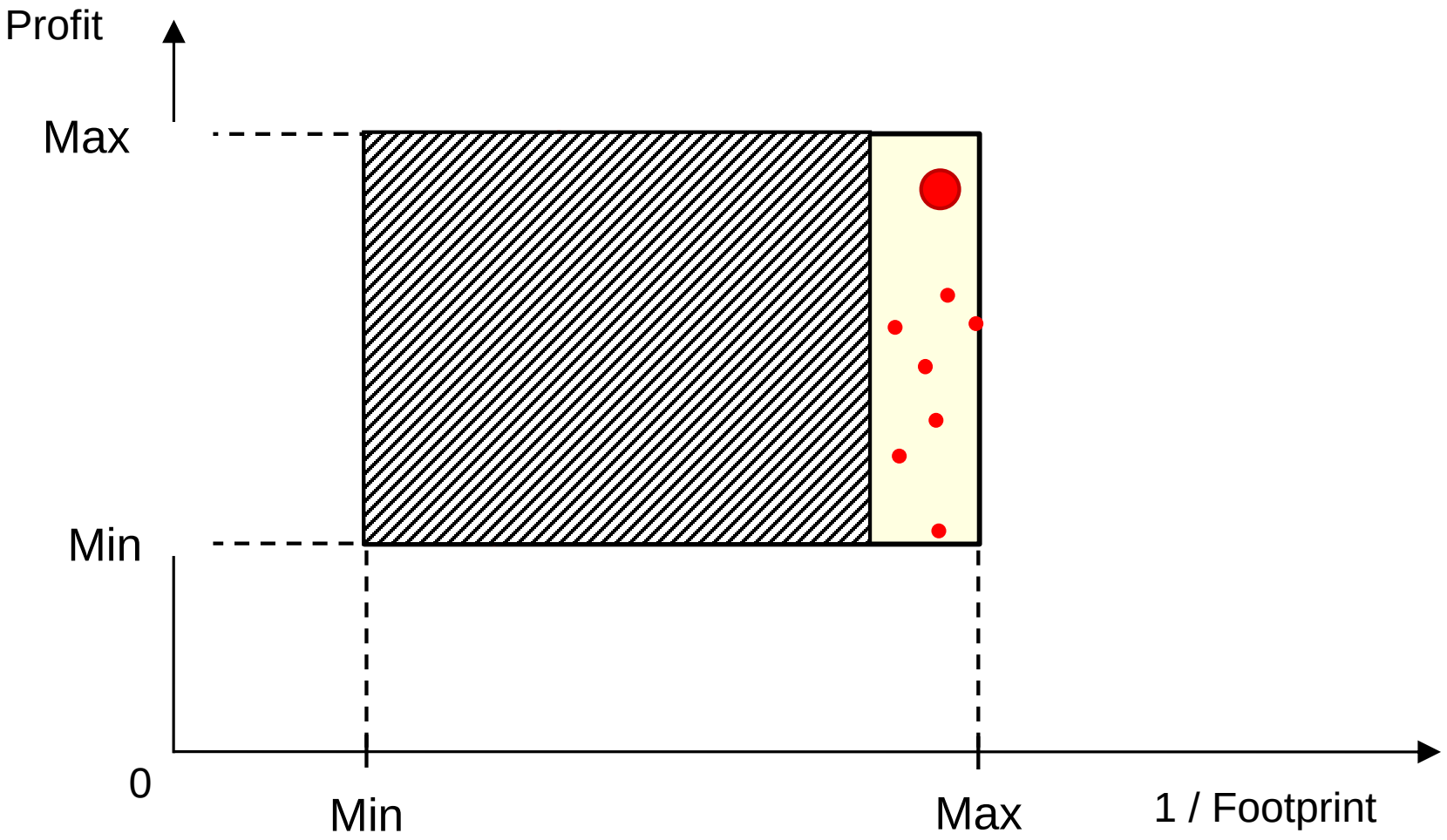
Sustainability map



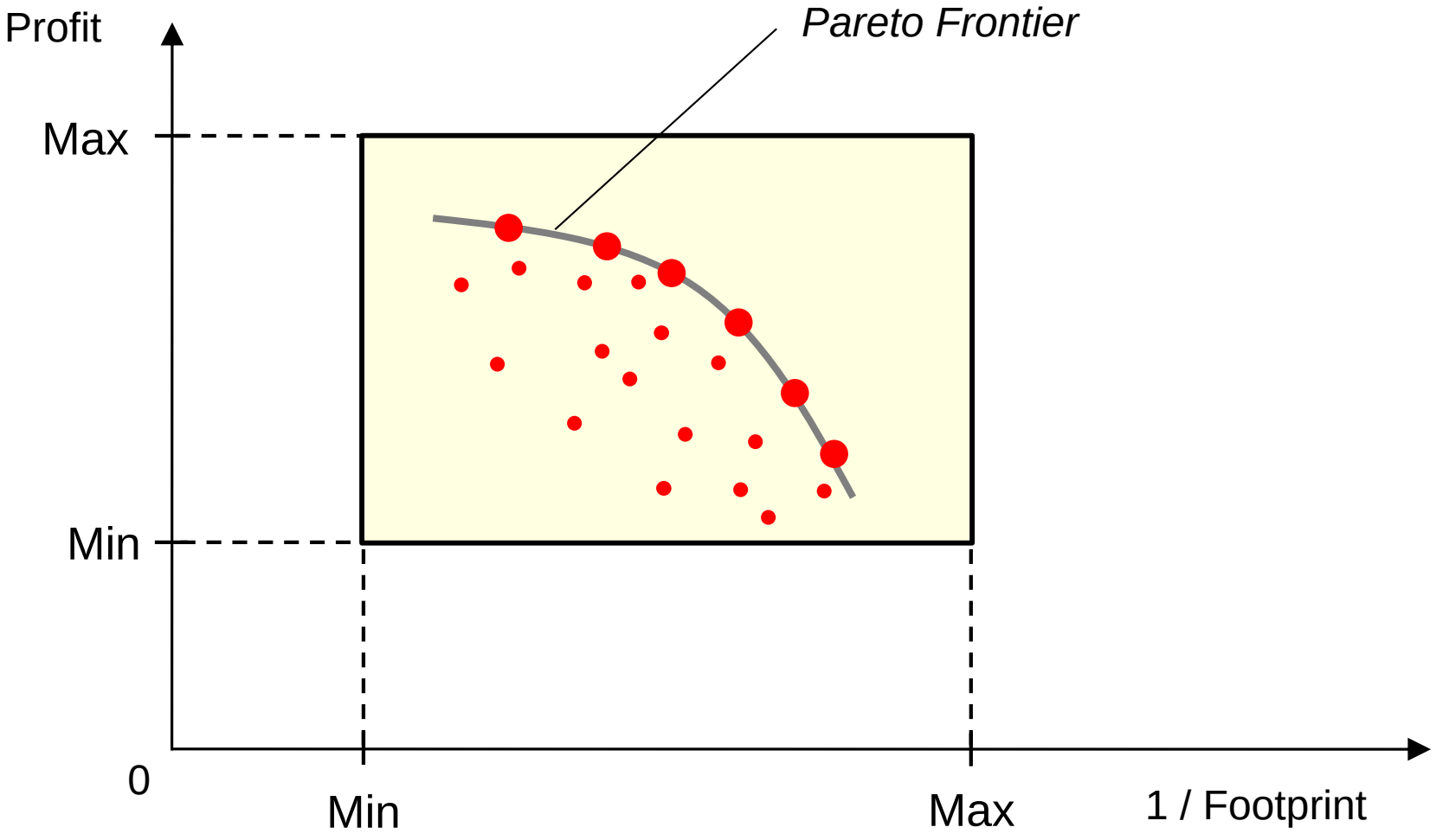
Selection: Constraint on profit



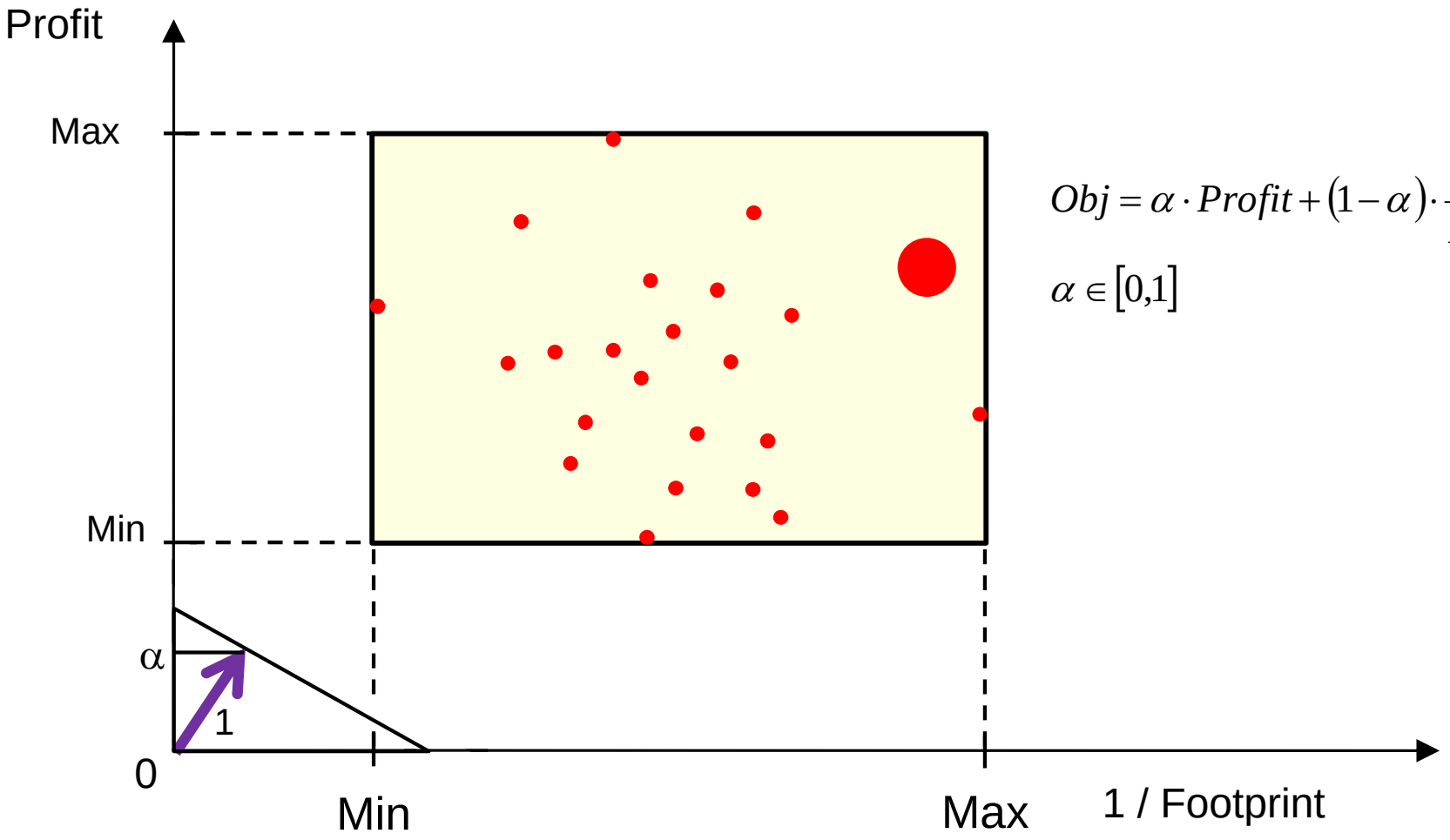
Selection: Constraint on sustainability



Selection: Multi-objective



Selection: Single objective



$$Obj = \alpha \cdot Profit + (1-\alpha) \cdot \frac{1}{Footprint}$$

$$\alpha \in [0,1]$$

P-graph framework

Sustainable supply-chain design via P-graph framework

Potential objective functions:

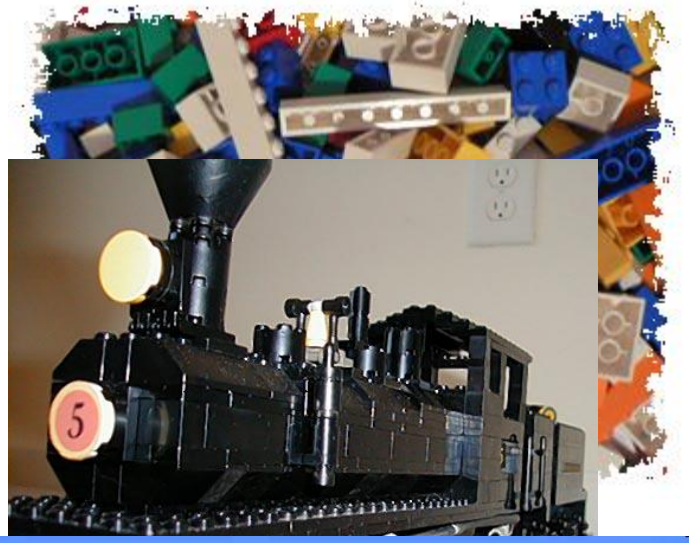
- profit
- ecological footprint
- exergy dissipation
- green net regional product
- ratio between renewable and total emergies

The P-graph framework:

- developed for process synthesis problems
- mathematically rigorous
- synthesize optimal and **n-best networks**

Building credible scenarios …

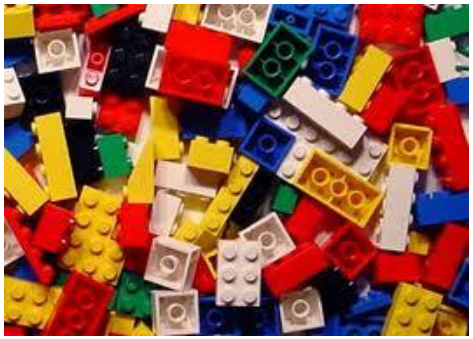
- Starting with **building blocks**…



- … create **comprehensive scenarios**…

- … that **help decision makers**!

Supply chain design as process synthesis



Building blocks

Industrial plants, works

Logistics

Molecular design

Renewable energy supply chain

The P-graph framework

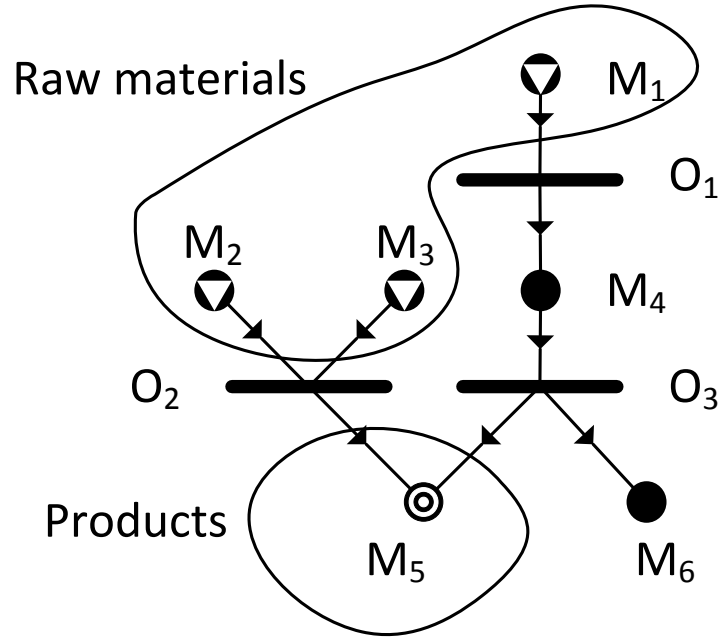
- Process systems engineering (PSE)
  - process design (flowsheeting)
    - **process synthesis**
    - process simulation (analysis)
  - process operation
    - scheduling
- The P-graph framework helps to address and solve process network synthesis problems

P-graph representation (Friedler and Fan, 1992)

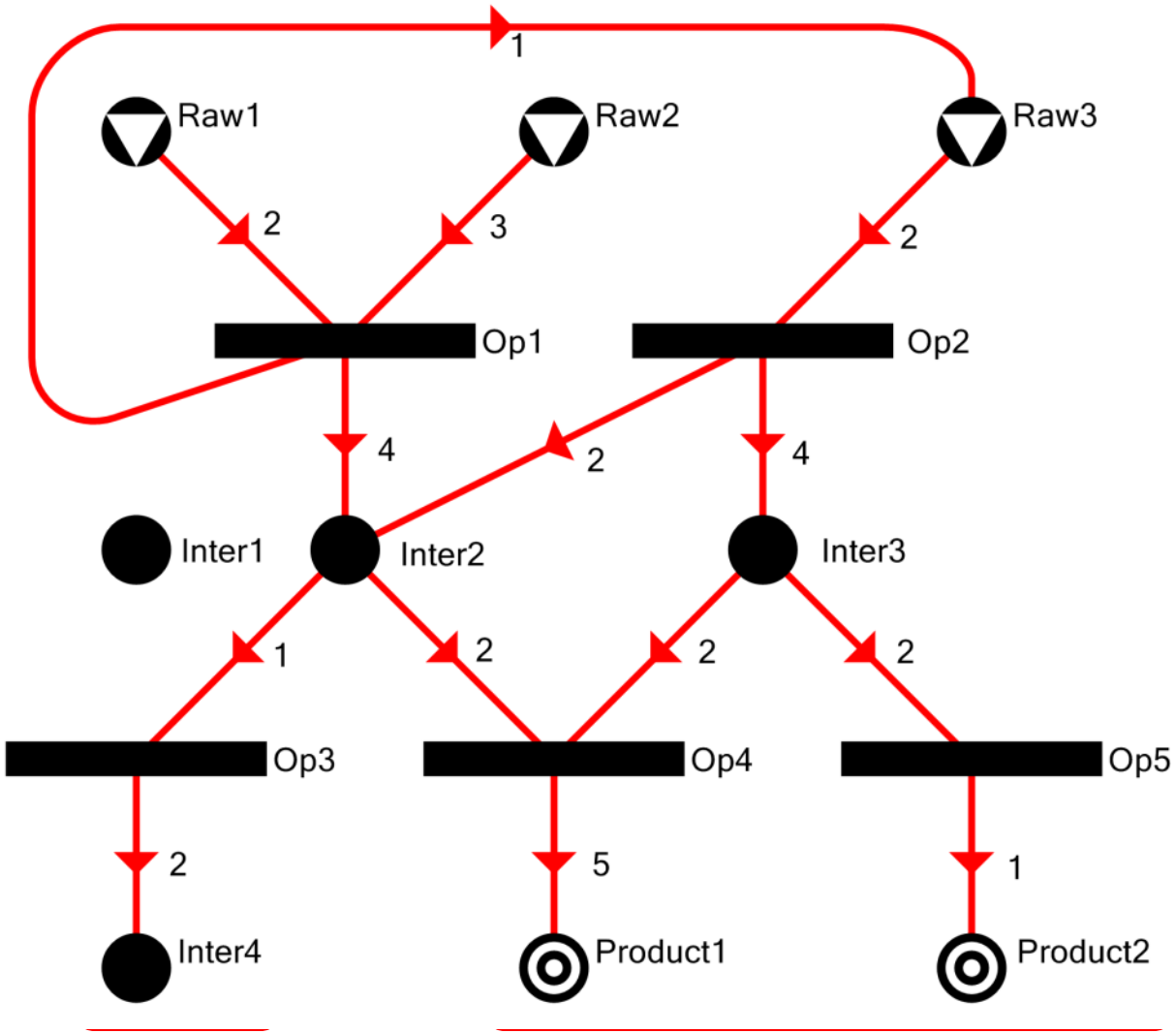
- Building blocks:
  - materials:
    - raw materials:
    - products:
    - intermediates:
  - operating units:

Synthesis

Process structure:



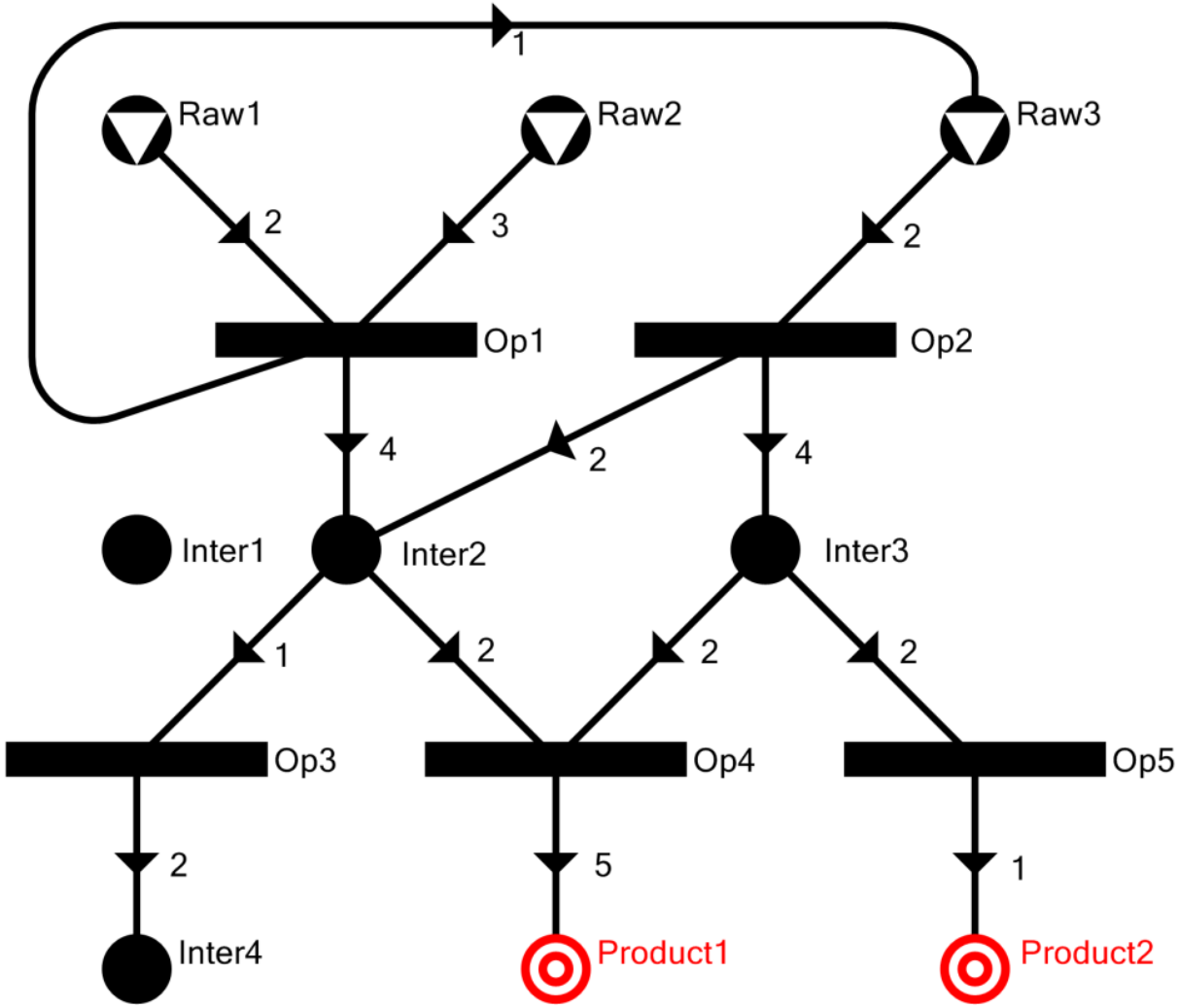
P-graph representation



Notations

- Nodes:
  - materials
    - raw materials
    - intermediate
    - products
  - operating units
- Arcs

P-graph representation

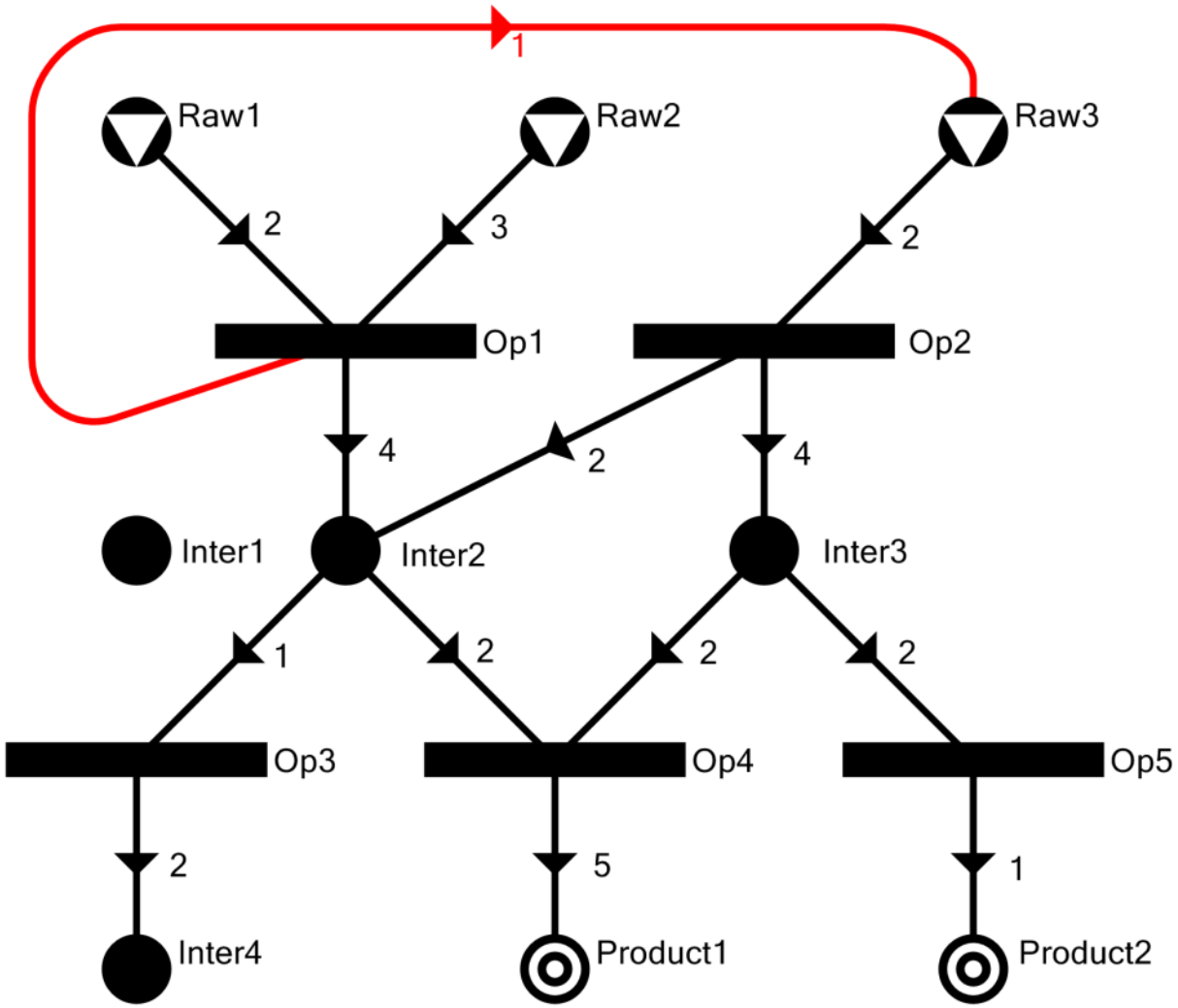


Axioms

(S1)

$P \subseteq m$

P-graph representation



Axioms

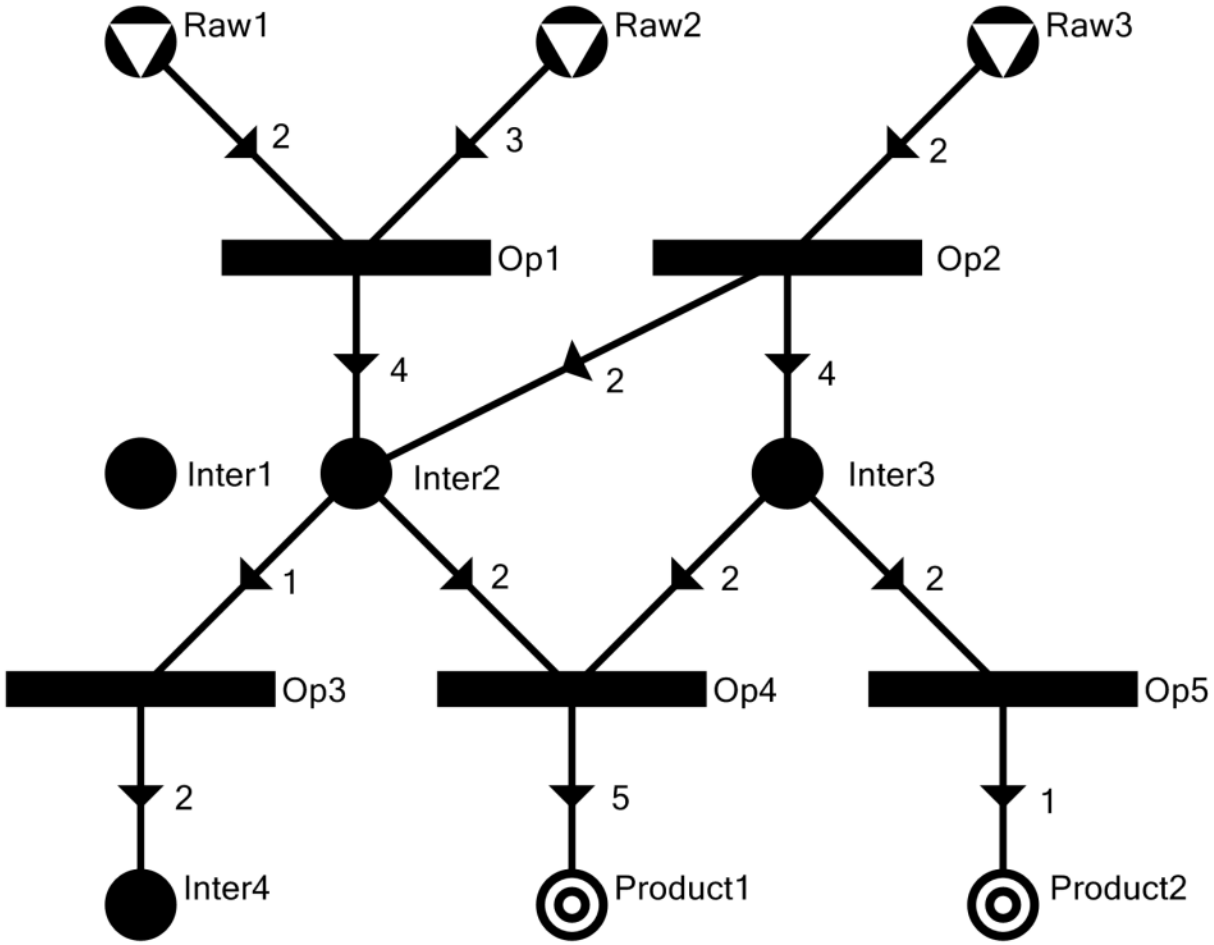
(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

P-graph representation



Axioms

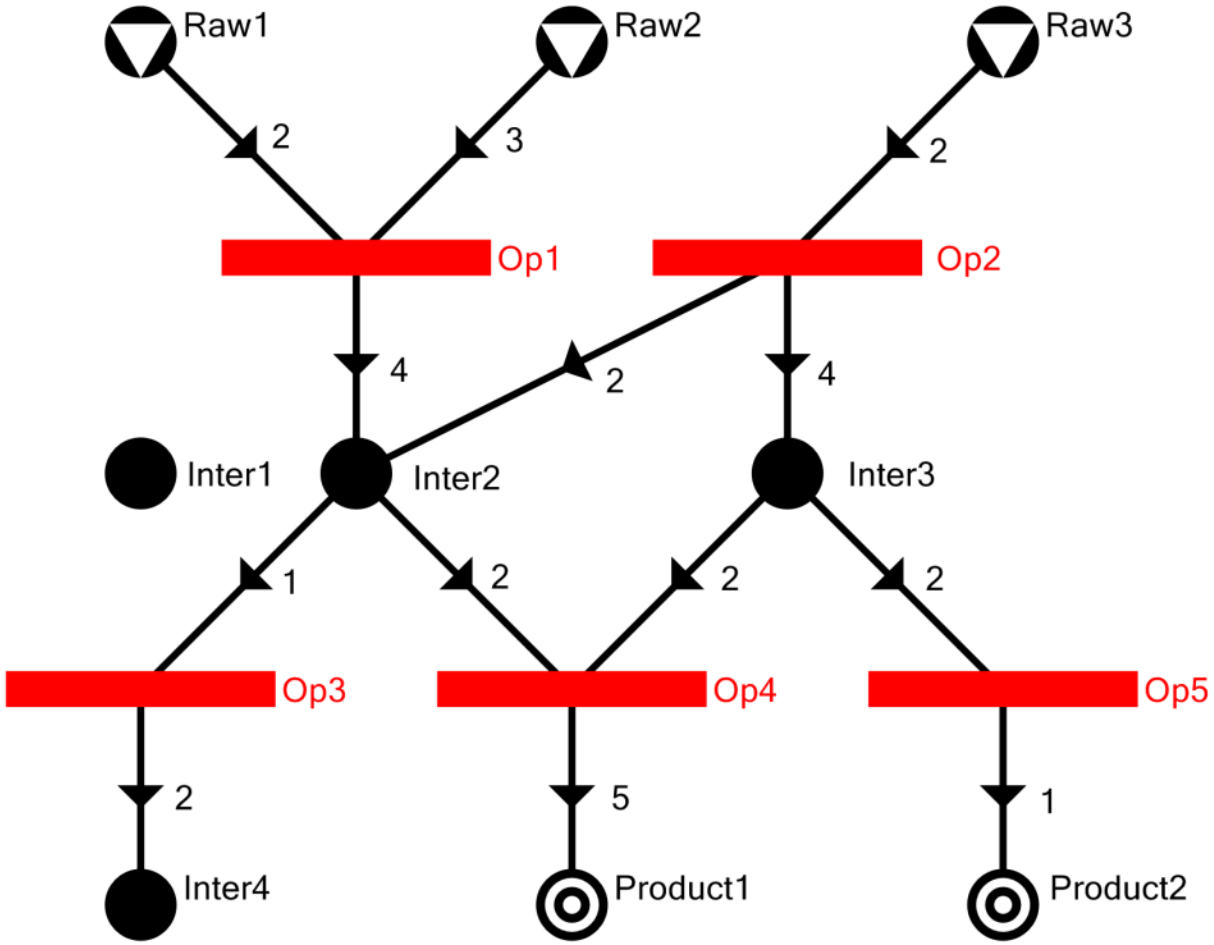
(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

P-graph representation



Axioms

(S1)

$P \subseteq m$

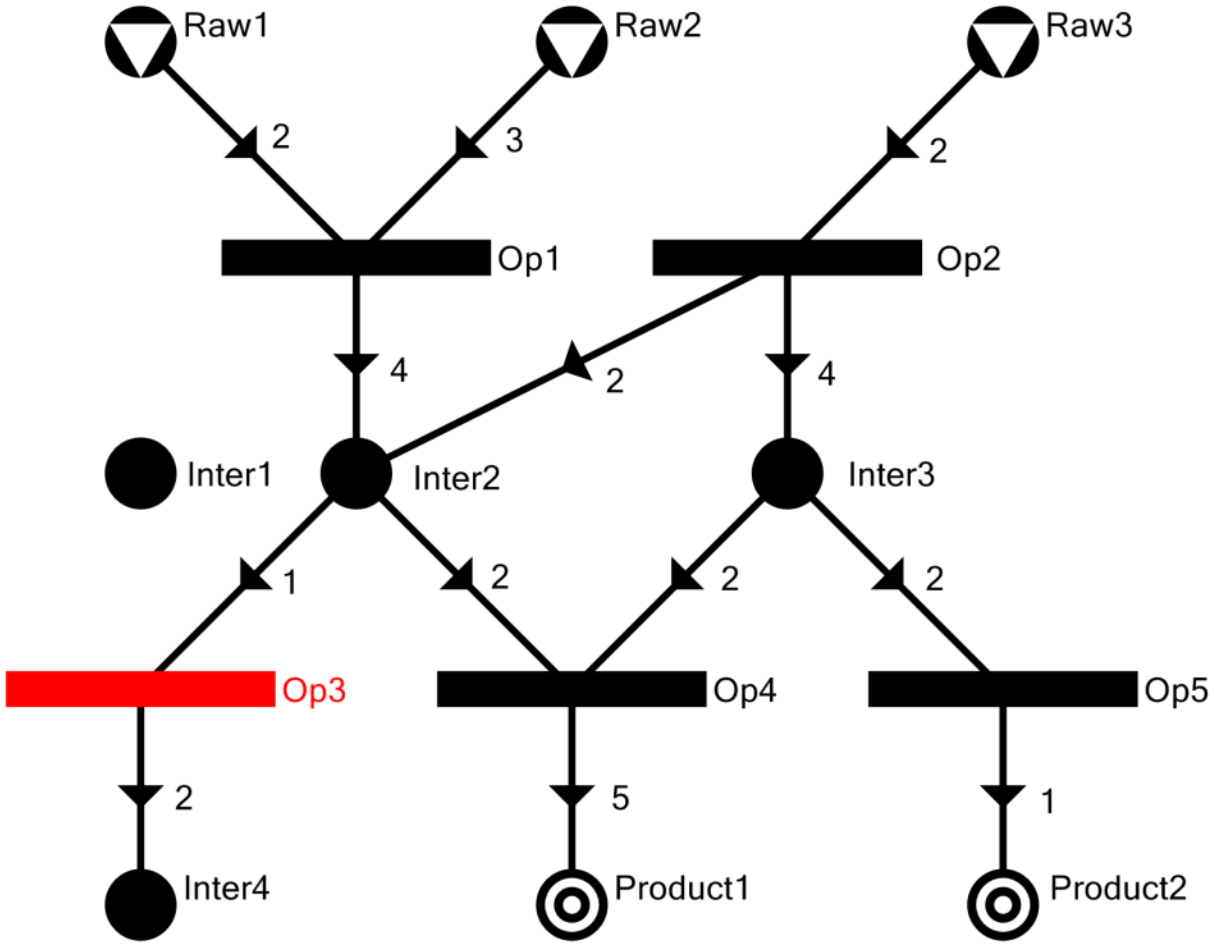
(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

(S3)

$o \subseteq O$

P-graph representation



Axioms

(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

(S3)

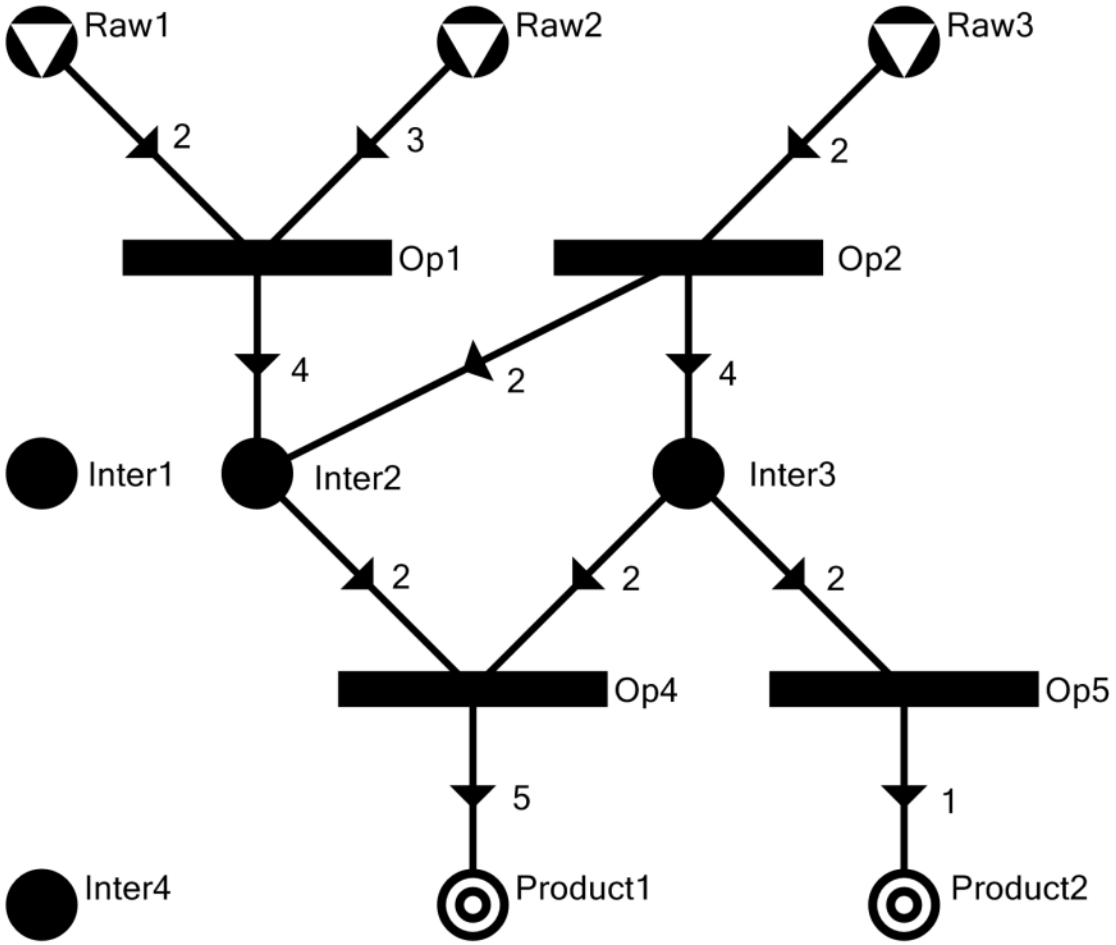
$o \subseteq O$

(S4)

$\forall y_0 \in o, \exists \text{ path } [y_0; y_1],$

where $y_1 \in P$

P-graph representation



Axioms

(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

(S3)

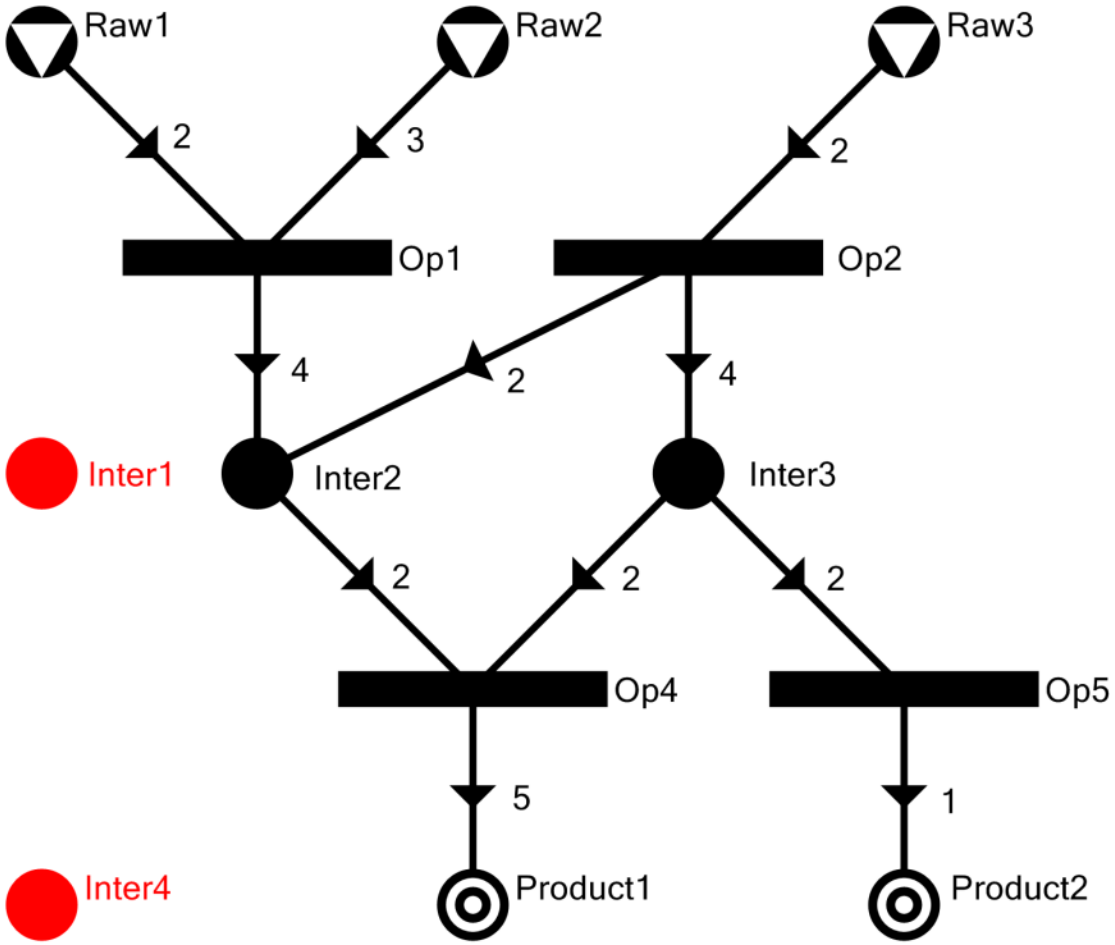
$o \subseteq O$

(S4)

$\forall y_0 \in o, \exists \text{ path } [y_0; y_1],$

where $y_1 \in P$

P-graph representation



Axioms

(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = 0$

(S3)

$o \subseteq O$

(S4)

$\forall y_o \in o, \exists$ path $[y_o; y_1]$,

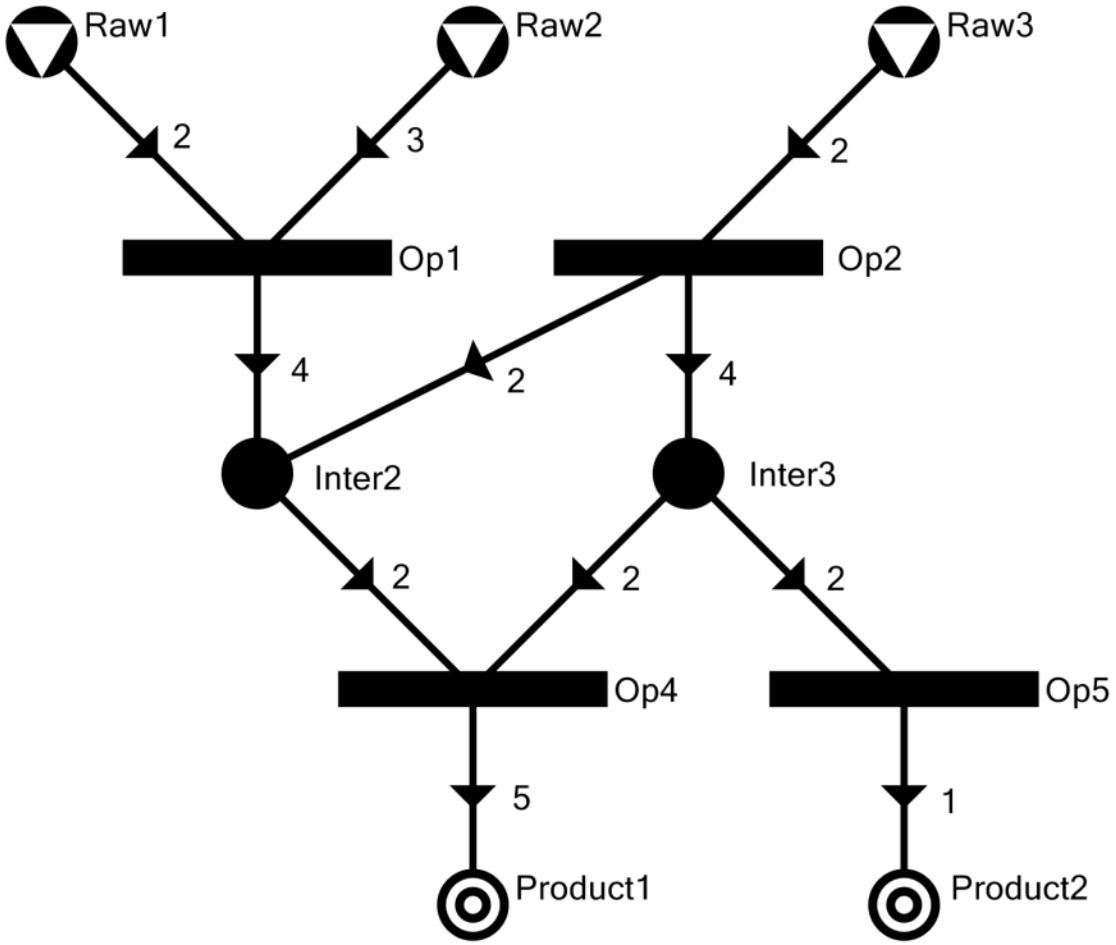
where $y_1 \in P$

(S5)

$\forall x \in m, \exists (\alpha; \beta) \in o;$

$x \in \{\alpha \cup \beta\}$

P-graph representation



Axioms

(S1)

$P \subseteq m$

(S2)

$x \in R \Leftrightarrow \forall x \in m, d^{-}(x) = o$

(S3)

$o \subseteq O$

(S4)

$\forall y_o \in o, \exists$ path $[y_o; y_1]$,

where $y_1 \in P$

(S5)

$\forall x \in m, \exists (\alpha; \beta) \in o;$

$x \in \{\alpha \cup \beta\}$

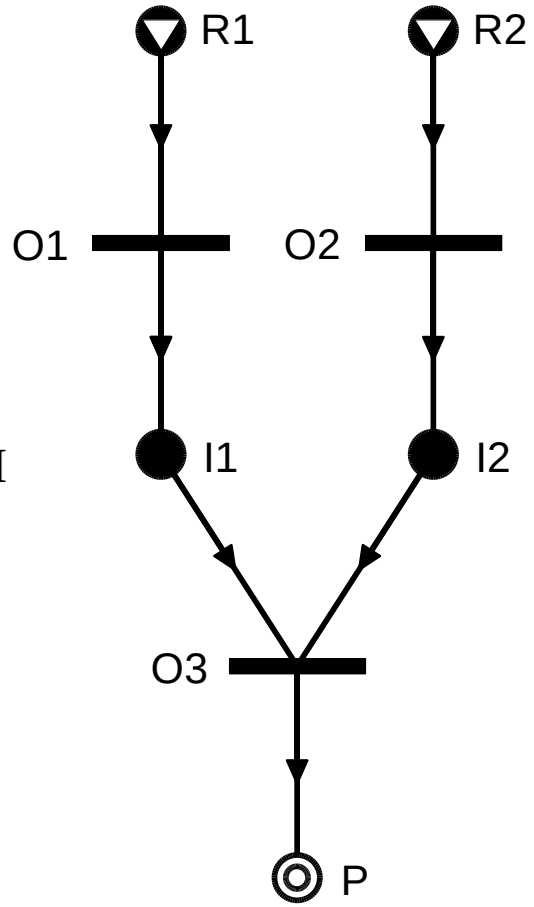
P-graph methodology

$$\min \sum_{o \in \mathrm{O}} \left(A_o \cdot y_o + B_o \cdot x_o \right) + \sum_{m \in \mathrm{R}} \left(C_m \cdot \sum_{\{o \in \mathrm{O} : (m,o) \in \mathrm{A}\}} \left(R_{o,m} \cdot x_o \right) \right)$$

s.t.

$$\sum_{\{o \in \mathrm{O} : (m,o) \in \mathrm{A}\}} \left(R_{o,m} \cdot x_o \right) + P_m \leq \sum_{\{o \in \mathrm{O} : (o,m) \in \mathrm{A}\}} \left(R_{o,m} \cdot x_o \right) + S_m \quad \forall m \in \mathrm{M}$$

$$x_o \leq y_o \cdot M \quad \forall o \in \mathrm{O}$$



P-graph algorithms

Maximal Structure Generator (MSG)

- maximal structure contains all potential solution structure
- it works in polynomial time

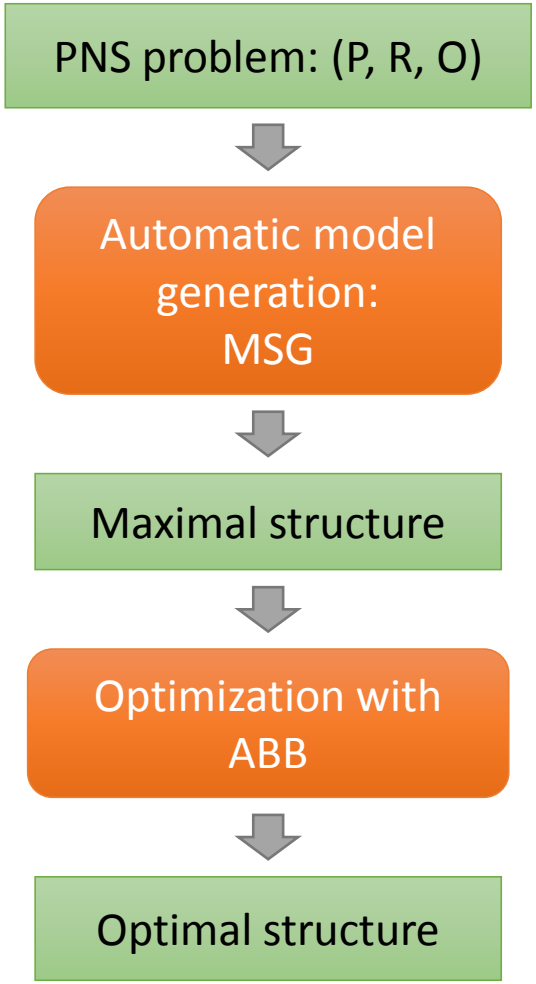
Solution Structures Generator (SSG)

- creates all the solutions one by one
- applicable for small problems

Accelerated Branch-and-Bound (ABB)

- locates the best or n-best optimal structure
- accelerates using the structural properties of P-graph

Process synthesis by P-graph framework



MINLP vs. P-graph framework

	MINLP	P-graph framework
Problem given by	Variables, Constraints	Raw materials, products, operating units
Generation of the math. model	Manual	Automatic
Structural properties of process-networks	Hidden in the math. model	Exploited
Number of solutions	Single	Multiple
Handling special constraints	Can be incorporated to the math. model	May require modifications of the model generator and solver

The metrics of sustainability

Sustainability Metrics

Metric		System Property
Ecological Footprint	↔	Ecological Burden
Exergy	↔	Energy Efficiency
Emergy	↔	Energy Resources
Green Net Prod	↔	Economy
Fisher Information	↔	System Order

Sustainable Process Index (SPI)

- SPI is a measure developed to evaluate the viability of processes under sustainable economic conditions
- The concept of the SPI is based on the assumption that in a truly sustainable society the basis of economy is the sustainable flow of solar exergy
- SPI **evaluates the areas needed** to provide the raw materials and energy demands and to accommodate by-product flows from a process in a sustainable way

The algorithm

- The total area

$$A_{tot} = A_R + A_E + A_I + A_{ST} + A_D \quad [m^2]$$

Tot	total
R	raw material
E	energy
I	infra structure
ST	staff
D	dissipation

- The specific area

$$a_{tot} = A_{tot}/N_S \quad [m^2a/service]$$

Ns	number of services provided per year

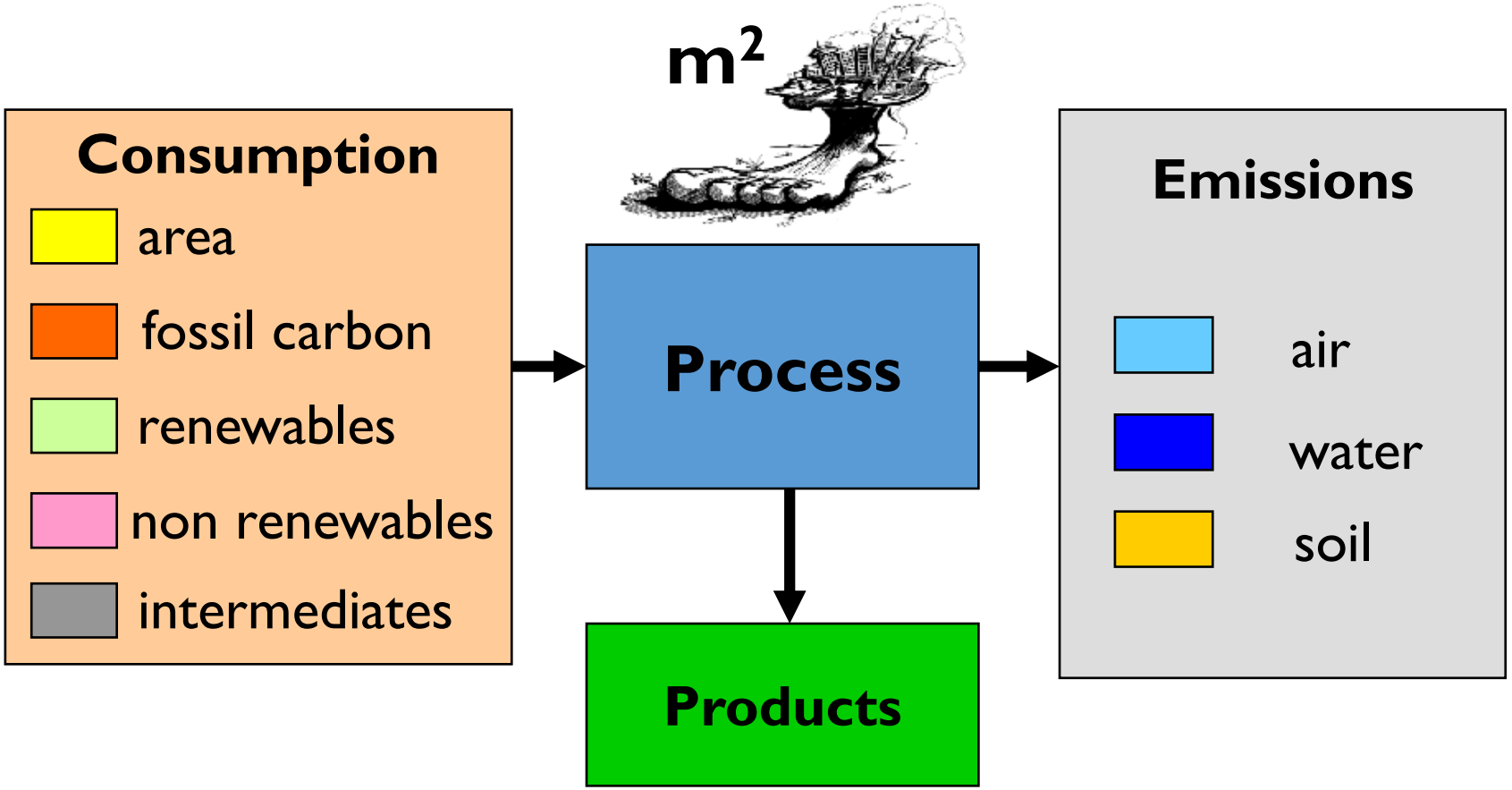
- The SPI

$$SPI = a_{tot}/a_{in}$$

(Sometimes a_{tot} is termed as SPI as well)

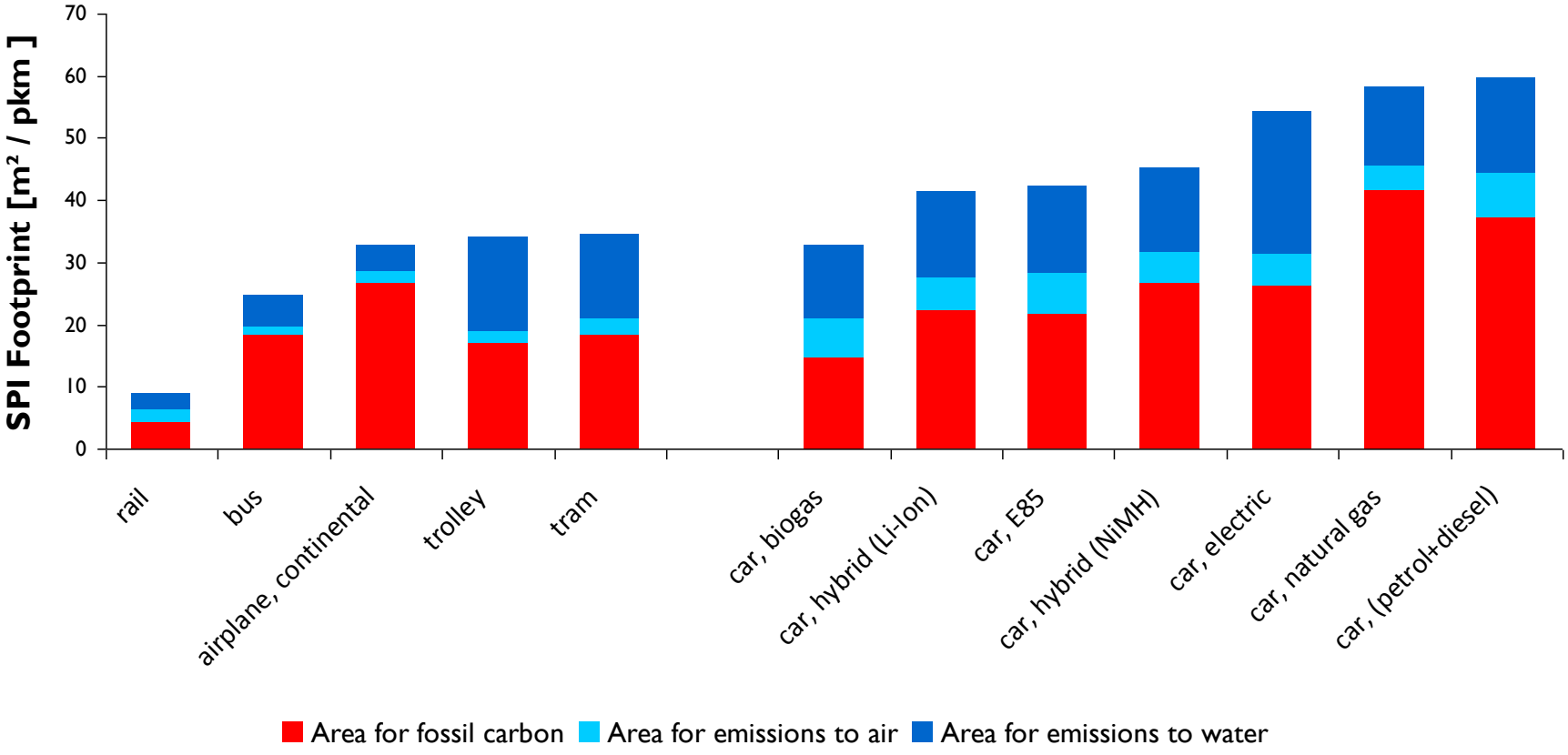
ain	Area per inhabitant

Calculating the SPI

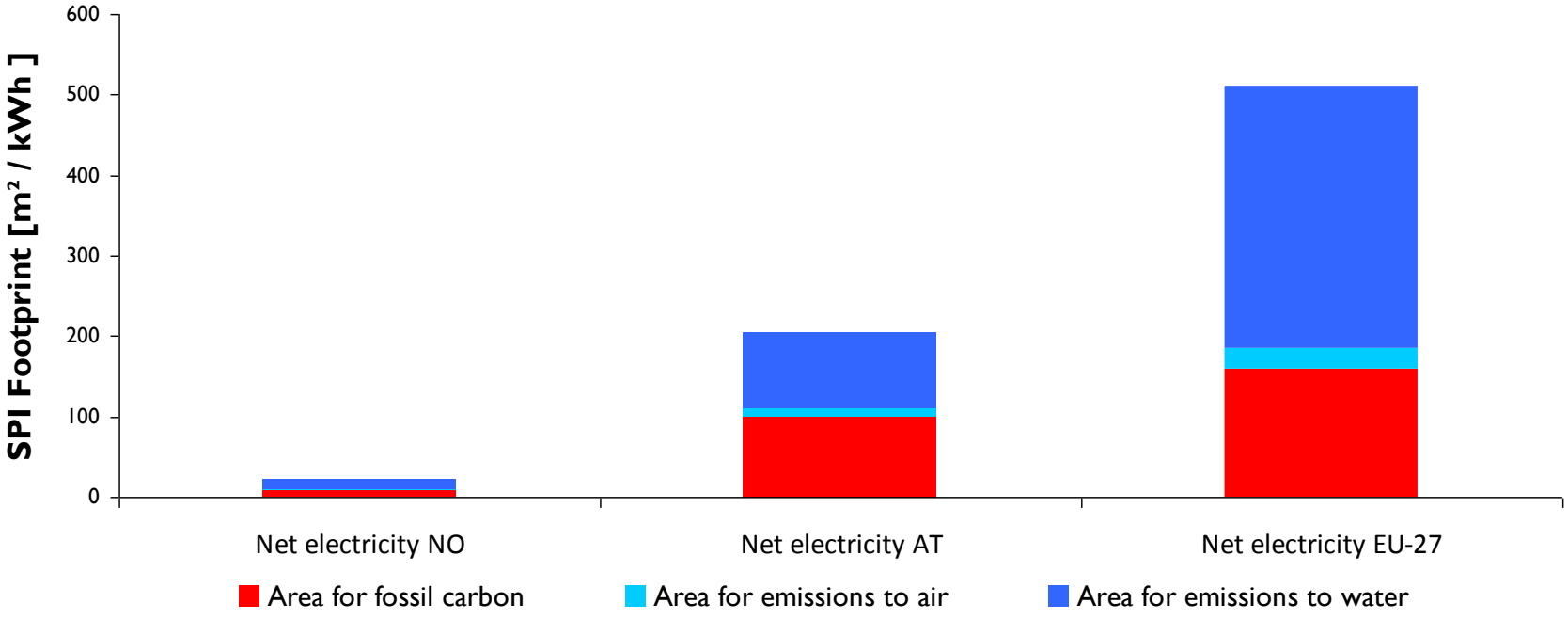


SPI results

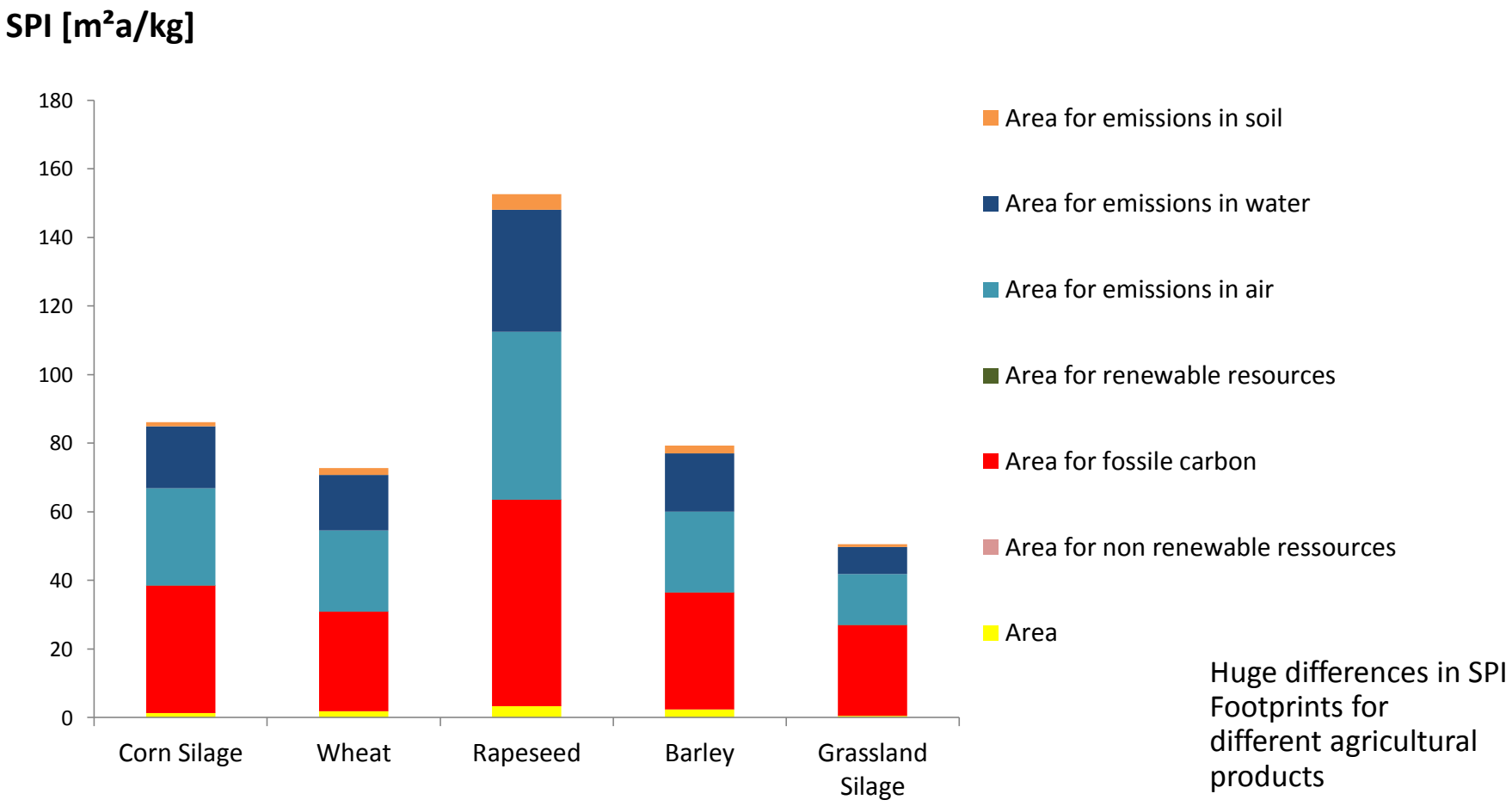
- Advantage for public transport



Country dependent SPI per kWh of electricity

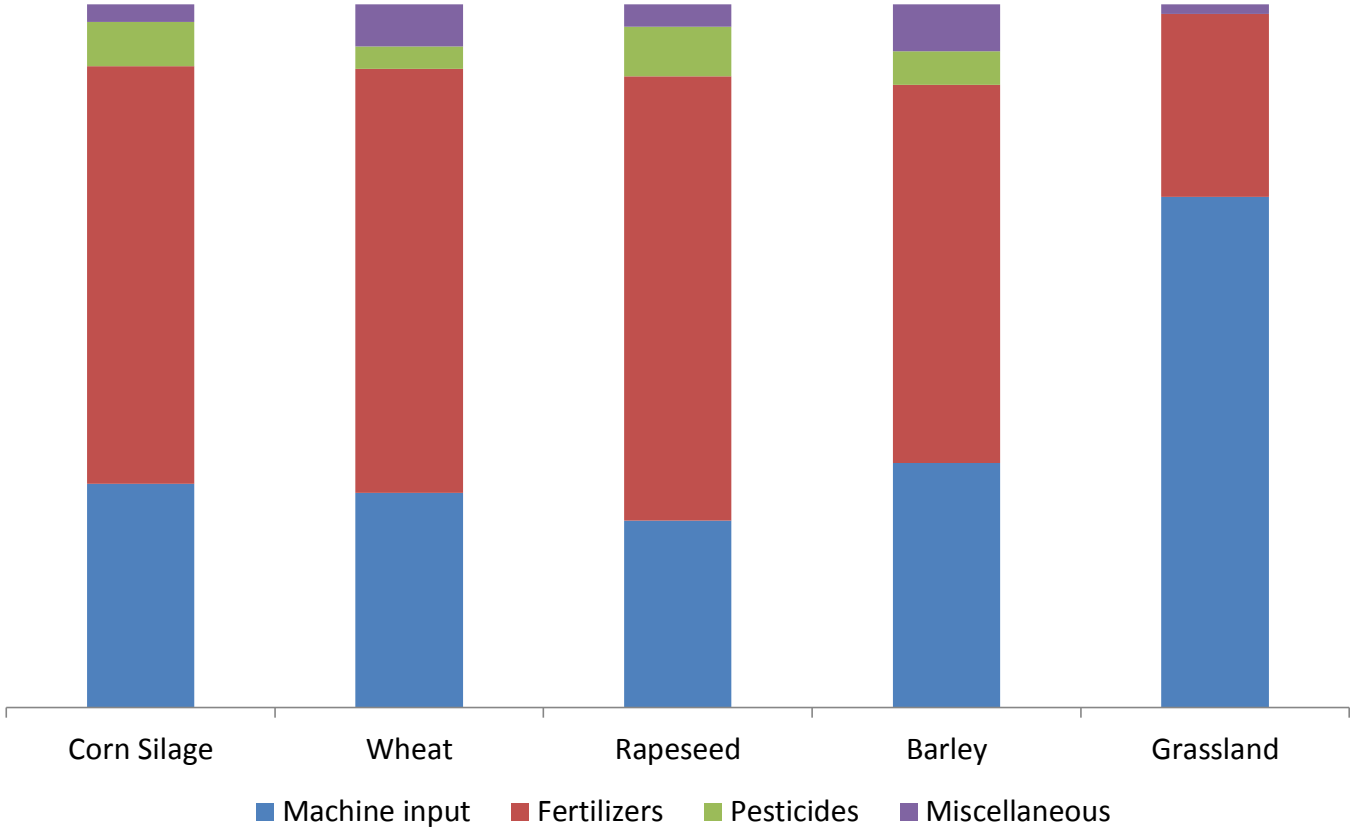


Ecological Footprint of agricultural base products



Huge differences in SPI Footprints for different agricultural products

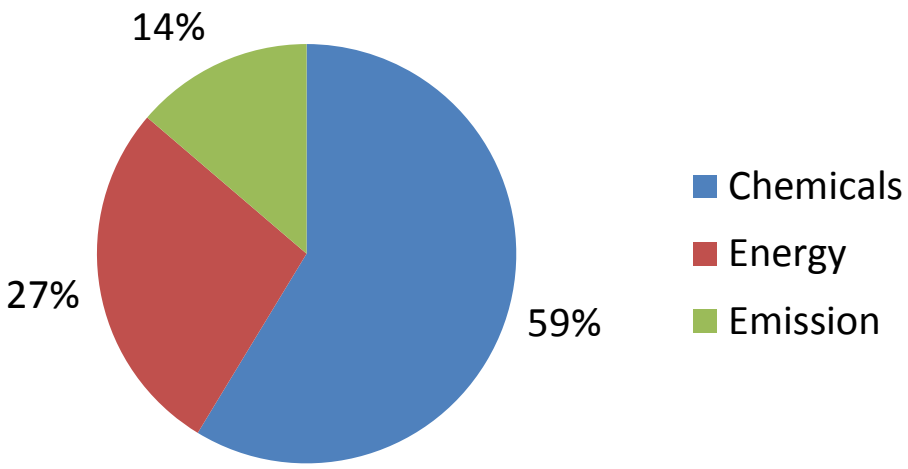
Ecological Footprint of agricultural base products



High impacts because of tractors and fertilizers

Ecological Footprint agricultural machines and fertilizers

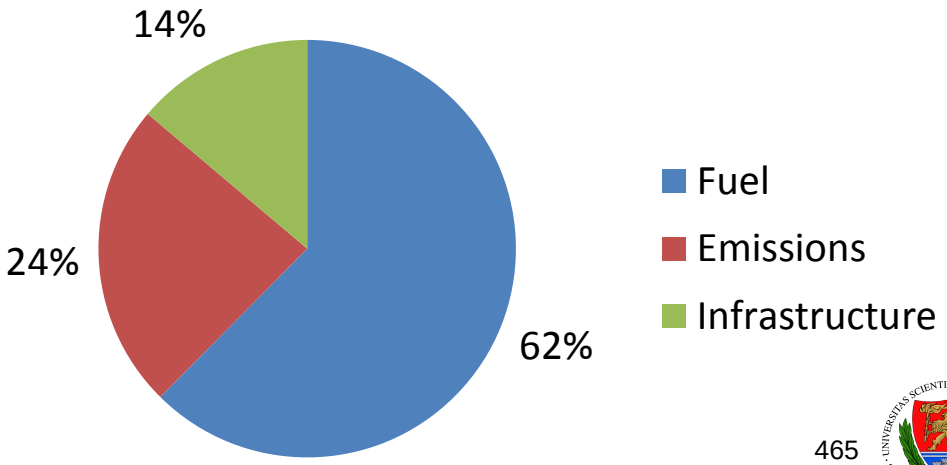
SPI for N-Fertilizer Ammonium nitrate phosphate



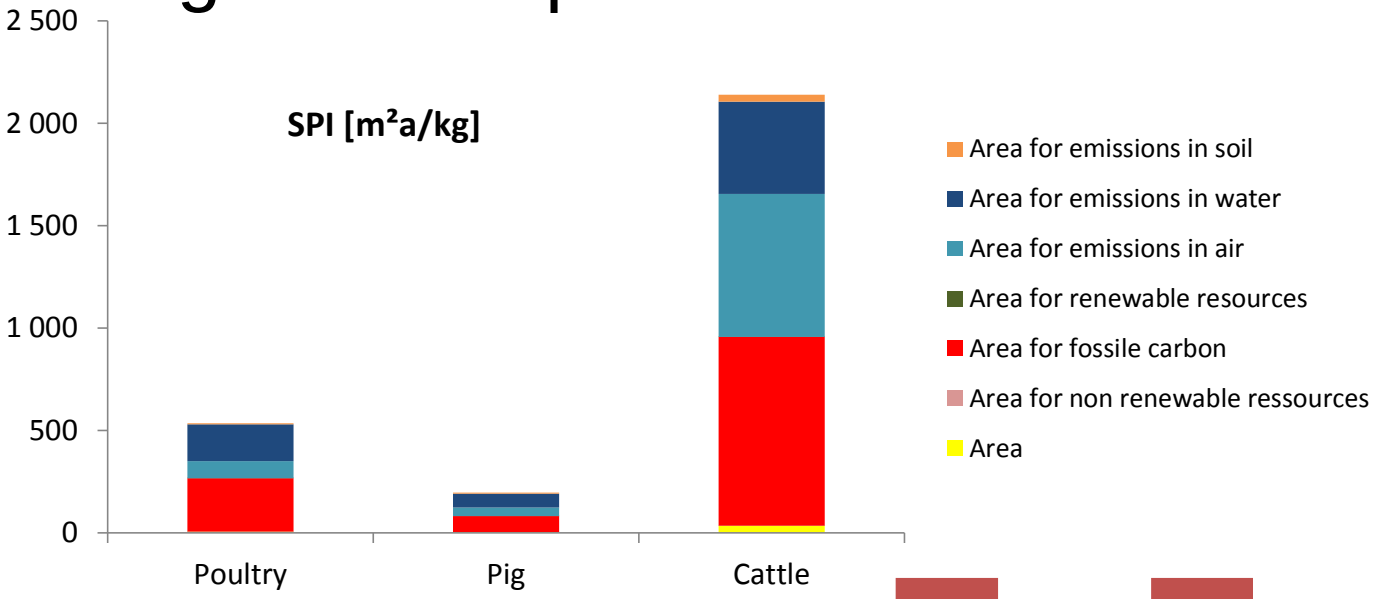
Industrial chemical production, energy provision and emission as main contributions to the Footprint for 1 kg of mineral fertilizer

Fuel consumption and emissions are mainly responsible for the impact of agricultural machines

SPI for Tractor (<70KW), normal workload

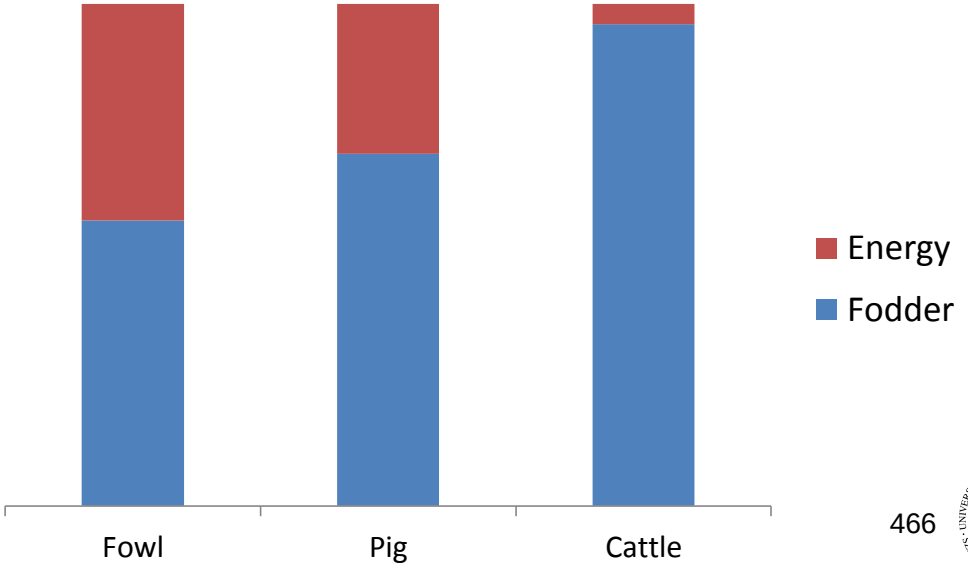


Ecological Footprint of „higher valued“ agricultural products



- Huge differences in meat production
- Yield in cattle production is very low (50 - 60 %)

Fodder production has already an accordingly high footprint which is „transferred“ to the final meat product



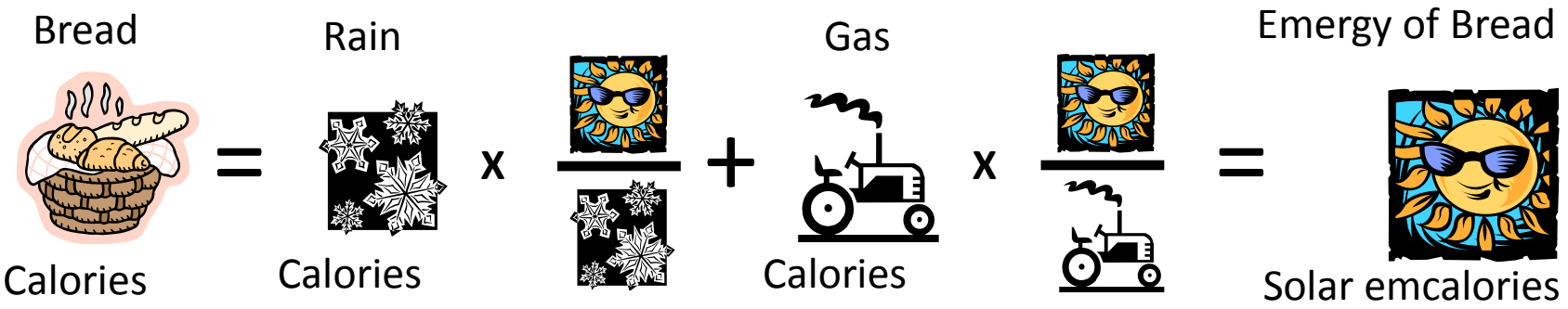
Exergy: Available Energy

- Exergy: thermodynamic work (in Joules) that can be done by bringing energy or mass into equilibrium with environment
  - a cup of hot tea has more exergy on the north pole than in a desert

- **Sustainability Criteria:** minimize exergy losses during processing
  - dEx/dt ≥ 0

Emergy

- Emergy: the sum of all different kinds of energy previously used (directly and indirectly) to make a product

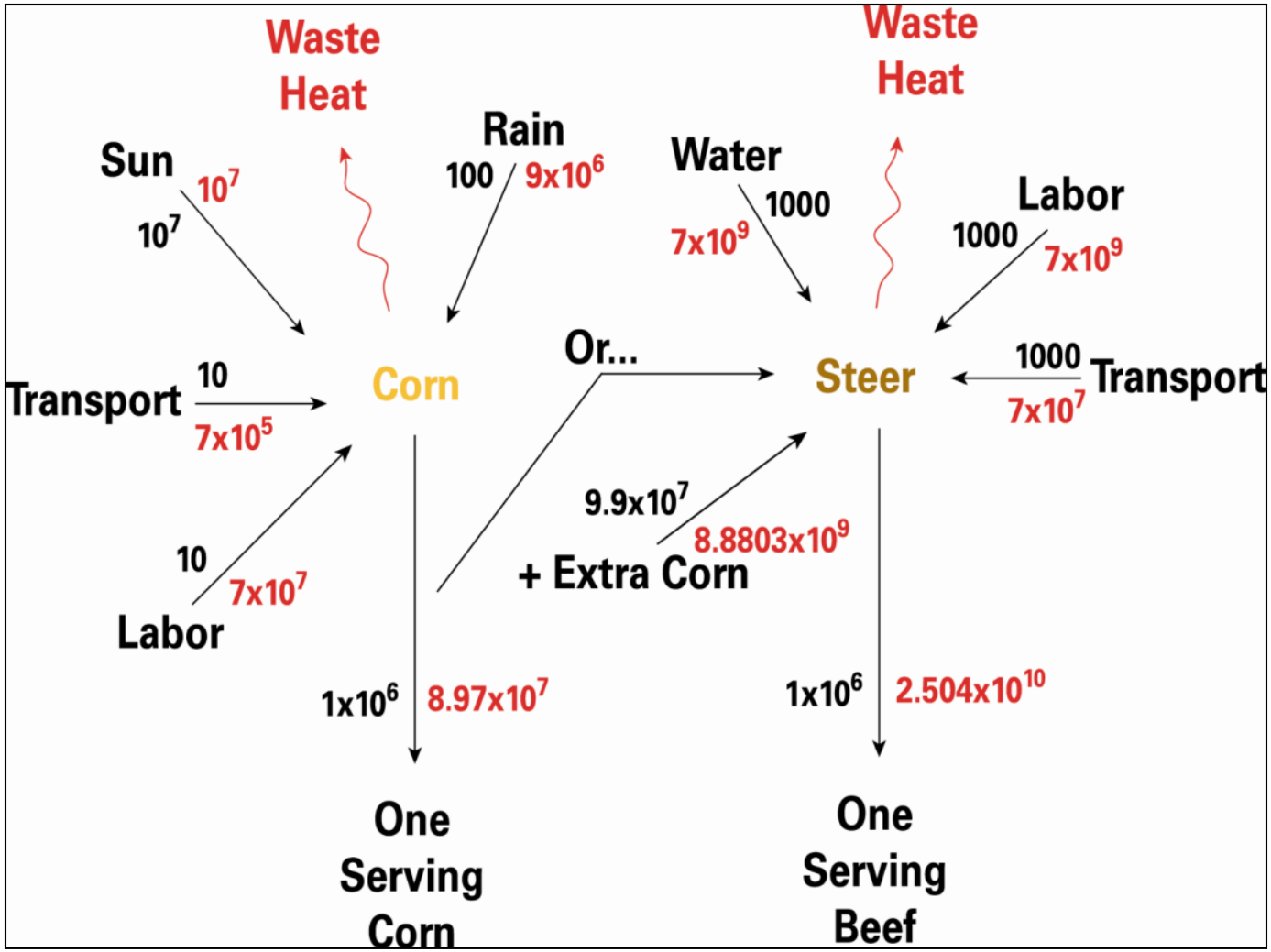


Emergy: Energy Resources

- Emergy: the energy resources (in solar joules) invested by the environment in an operation or in a product

- **Sustainability Criteria:**
  - (total emergy of inputs) → minimum
  - (renewable emergy use) / (total emergy use) → 1

Exergy vs. Emergy: Corn vs. Beef



Source: Mayer, A.L., Thurston, H.W., Pawlowski, C.W., The Multidisciplinary Influence of Common Sustainability Indexes, Front. Ecol. Environ, 2, 419-426 (2004).

Ecological Footprint Basics

- Land is categorized int
  - arable land
  - forest land
  - pasture land
  - sea
  - energy land
  - built land

blog.lib.umn.edu/tupp0008/environment/2008/02/

Ecological Footprint Basics

- Ecological Footprint (demand) = land area required to meet level of consumption and waste generation by the human population
- Biocapacity (supply) = land area available to support the human population

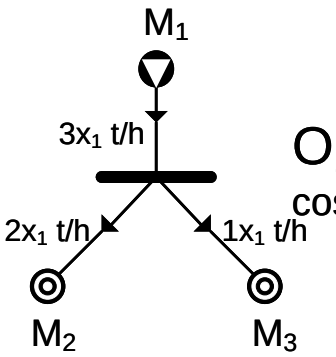
Ecological Footprint Basics

- Assumptions:
  - can track of resources and waste generated
  - resource and waste flows can be converted to land area
- **Sustainability Criteria:**
  - biocapacity is larger than ecological footprint
    - $B \geq EF$
  - ecological footprint does not increase and biocapacity does not decrease
    - $\Delta EF \leq 0$ & $B \geq EF$

Multi-period operating unit

P-graph representation

Definition of operating unit O_1

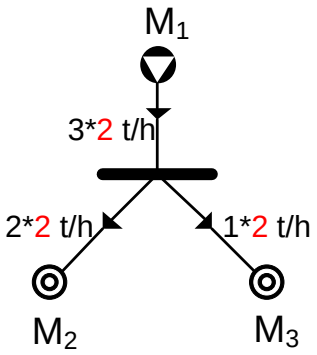


O_1
cost: $4x_1+20$ = 24 $/h

x_1: relative size
(dimensionless variable)

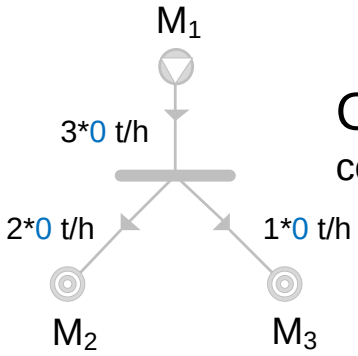
Two potential realizations of O_1

$x_1 = 2$



O_1
cost: 4*2+20
= 28 $/h

$x_1 = 0$

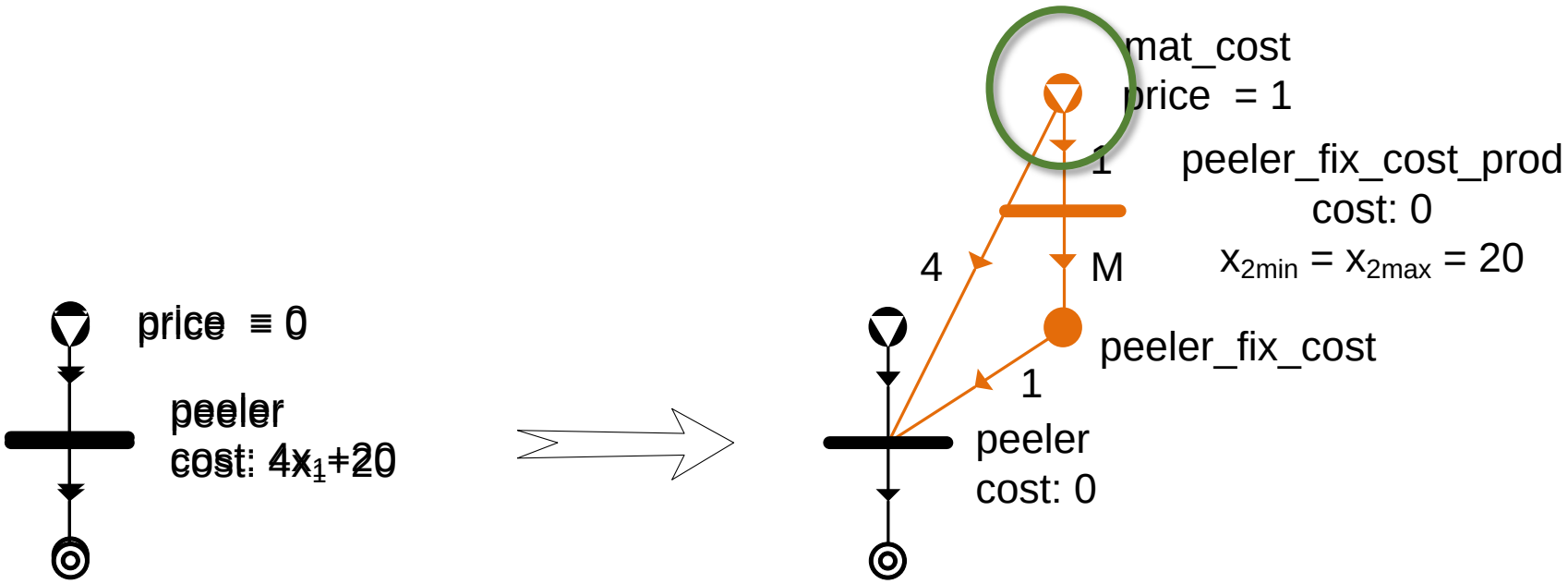


O_1
cost: 0 $/h

P-graph representation

- Definition of an operating unit: specifies the flowrates and the cost if the relative size is one
- **Relative size**: a multiplication factor, how much time bigger operating unit is needed than the definition
  - decision variable
  - the flowrates and the proportional part of the cost has to be multiplied accordingly
- If the relative size is 0 then the operating unit does not appear in the solution
  - the fix cost is also 0

Modeling total cost



$x_1 \equiv 10$
$4*10+20 \equiv 60$

$x_1 = 10, x_2 = 20$
$4*10 + 1*20 = 60$

Modeling total cost

- Old method: raw materials and operating units have associated cost
- New method: each operating unit consumes a new material termed mat_cost
  - the operating unit itself does not have cost
  - the cost of mat_cost is one

Modeling total cost

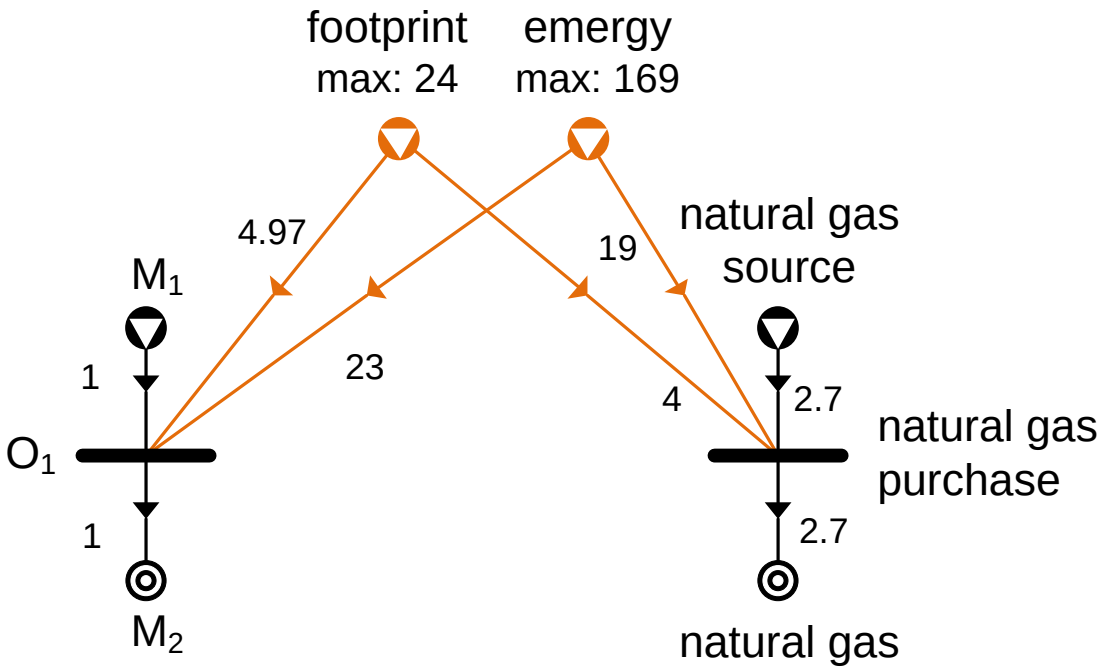
- Considering sustainability does not mean that cost is no longer an issue
- We may want to determine the best network in terms of footprint, but then cost can become a constraint
- A new material called, **mat_cost**, has to be introduced for the whole network
- A new operating, **peeler_fix_cost_prod**, unit and a new intermediate material, **peeler_fix_cost**, has to be introduced for each operating unit in the original network

Modeling total cost

- Old method: x = 10, cost = 4 * 10 + 20 = 60
- New method: x = 10
  - consumption of 40 unit from mat_cost
    - its price is also 40
  - 10 unit of the peeler_fix_cost is also needed
  - only peeler_fix_cost_prod can produce it, thus, this operating unit should be in the solution
  - if peeler_fix_cost_prod is in the solution then its relative size must be 20
    - it has to consume 20 unit from mat_cost
  - the cost is 60

Modeling of ecological footprint

Emergy, footprint representation



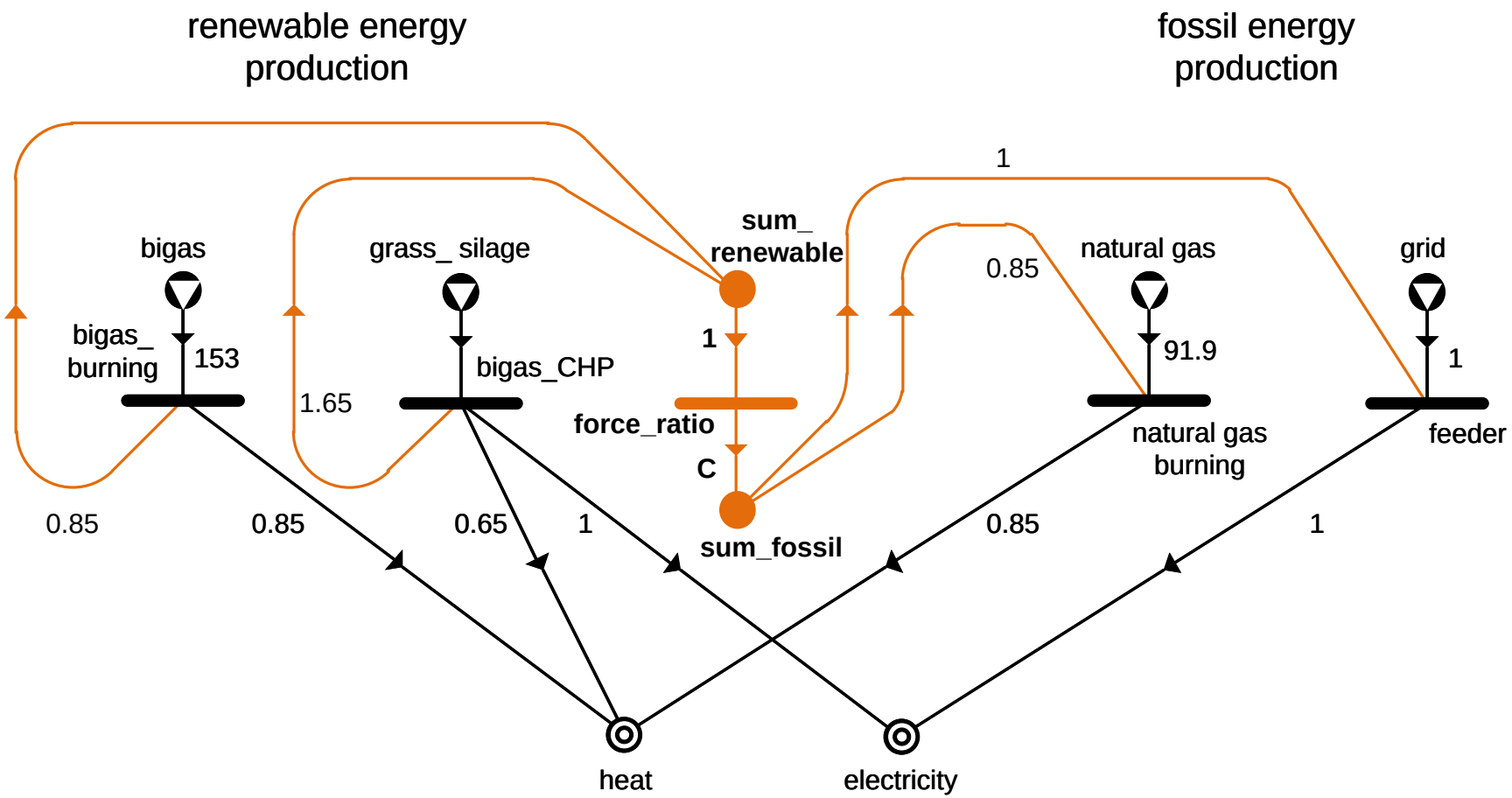
Emergy, footprint representation

- Ecological footprint and emergy are two sustainability metrics what we would like to incorporate into our model
- Both of these metrics are extensive
  - the footprint of a network can be calculated by summarizing the footprint of its components
- We would like to limit our search according to footprint, so the initial structure has to be transformed to impose this constraint

Emergy, footprint representation

- **New material nodes termed as footprint and emergy are introduced**
- These materials will be the inlet for each operating unit which represent some physical process
  - the new nodes have limits for maximum inlet flowrate
- If footprint belongs to a raw material then an operating unit is introduced to represent the purchase, and the footprint material will be the inlet of this unit

Ratio of renewable and fossil energy production



Ratio of renewable and fossil energy production

- We may search for solutions which are not worse than an already known solution in terms of the ratio of renewable and fossil energy production
  - for example, if our current design ensures the parity of renewable and fossil energy production, then we would not be satisfied with a solution where the ratio of the renewables is less than 50%
- A technique is proposed here to ensure this ratio
- The idea is to introduce material nodes: **sum_renewable** and **sum_fossil** to keep track the corresponding energy production

Ratio of renewable and fossil energy production

- These materials are connected through a new operating unit (force_ratio) which ensures the required constraint
  - if the parameter C is set to 1 then we can burn natural gas to create 1 MWh of energy only if the same amount is produced either from biogas or grass silage
- Material sum_renewable is produced when biogas is burned or when heat and electricity is produced from grass silage
- Material sum_fossil is consumed when natural gas is burned or when electricity is consumed from the grid
- It can happen that the ratio is worse than before because both type of energy production is reduced but the renewable production in larger volume

Illustrative example

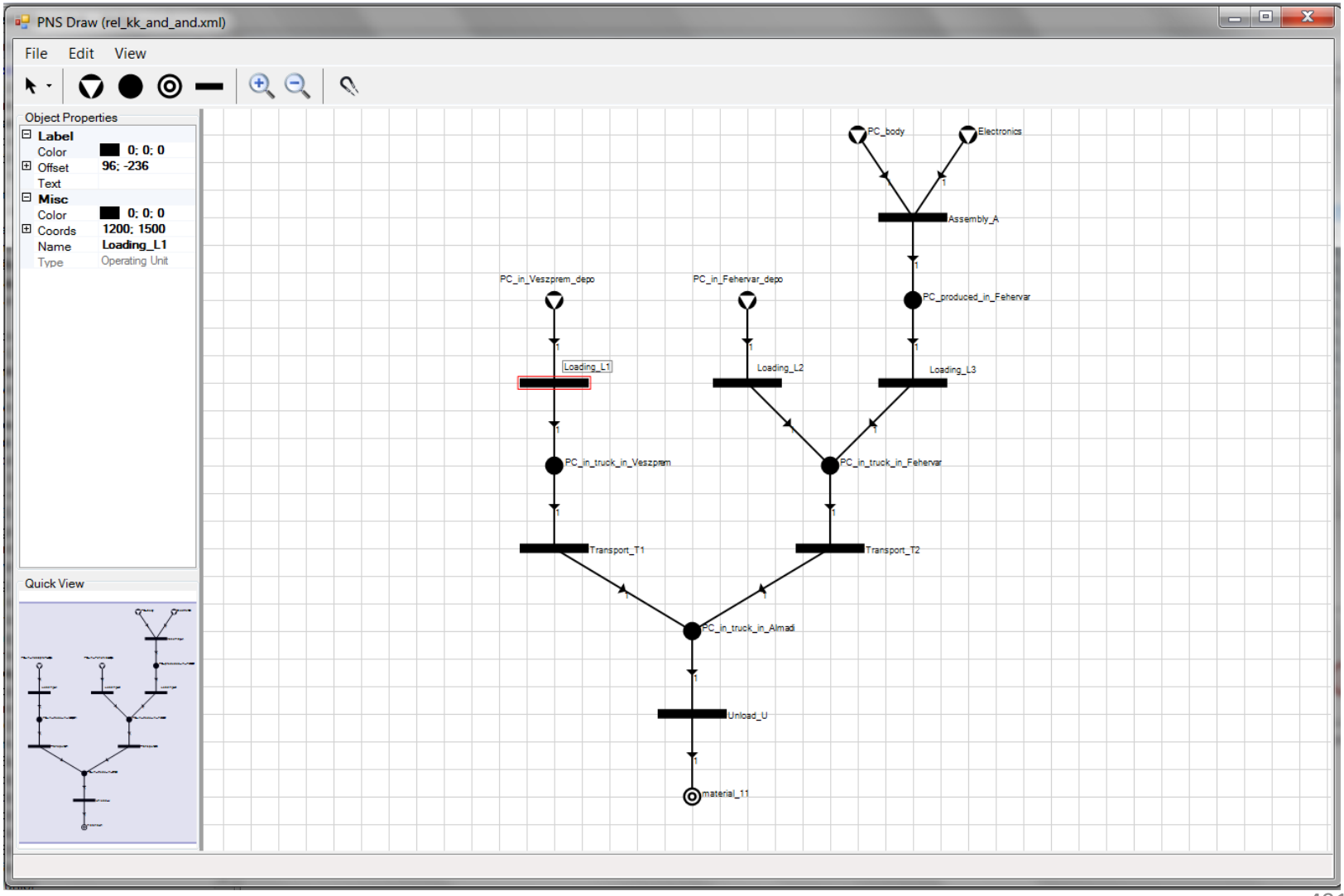
Assumptions

- The ecological footprint of the supply chain is dominated by the feedstocks and inputs
- The emergy footprint of the supply chain is also dominated by the feedstocks and inputs
- Methodology must be simple enough for wide application
- Supply chain is designed to produce 7.2 TJ/year of electricity and 18 TJ/year of heat

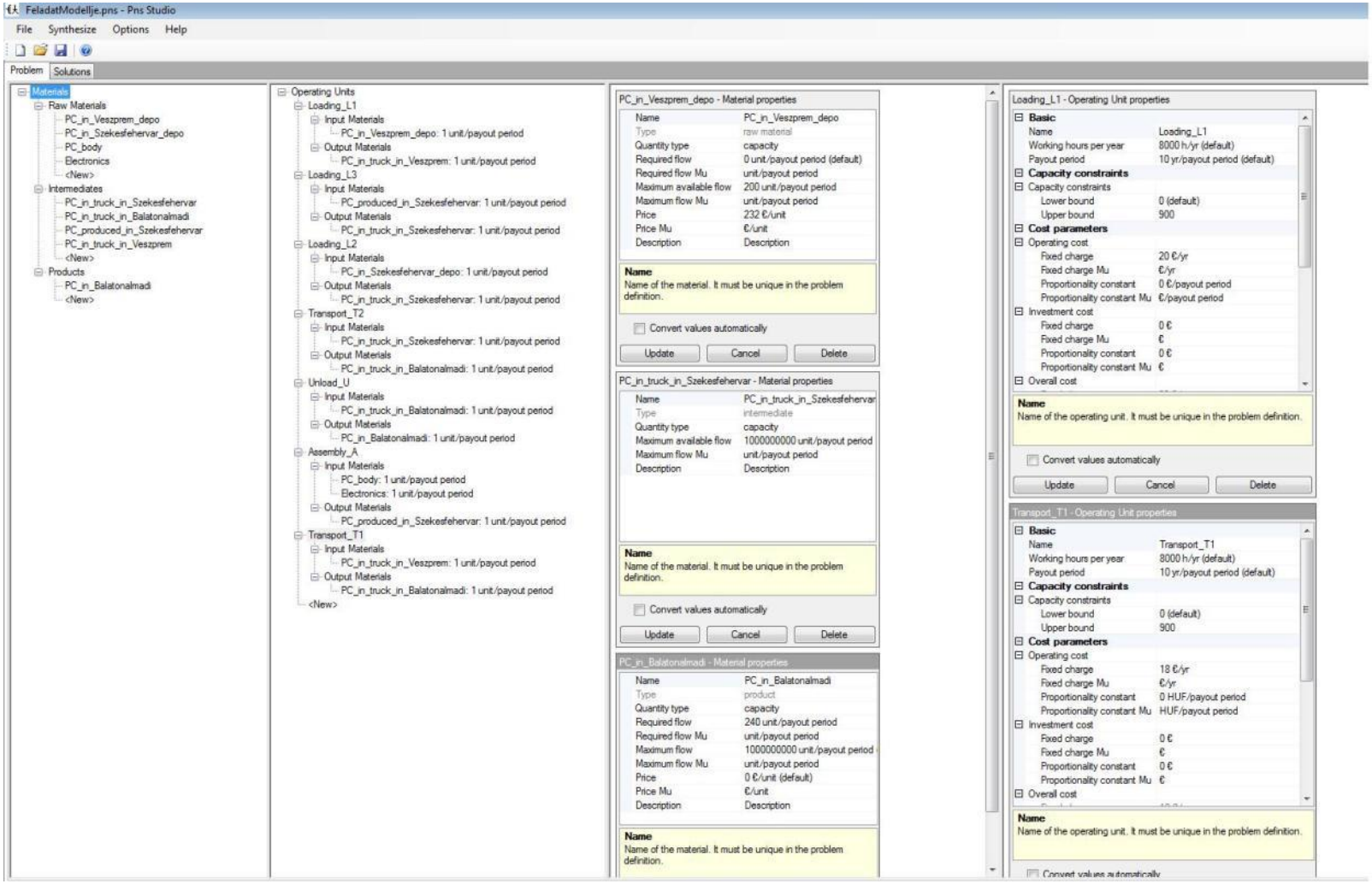
Applied tools

- PNS-Draw: graphically depict P-graphs
- PNS-Studio: solve and export synthesis problems

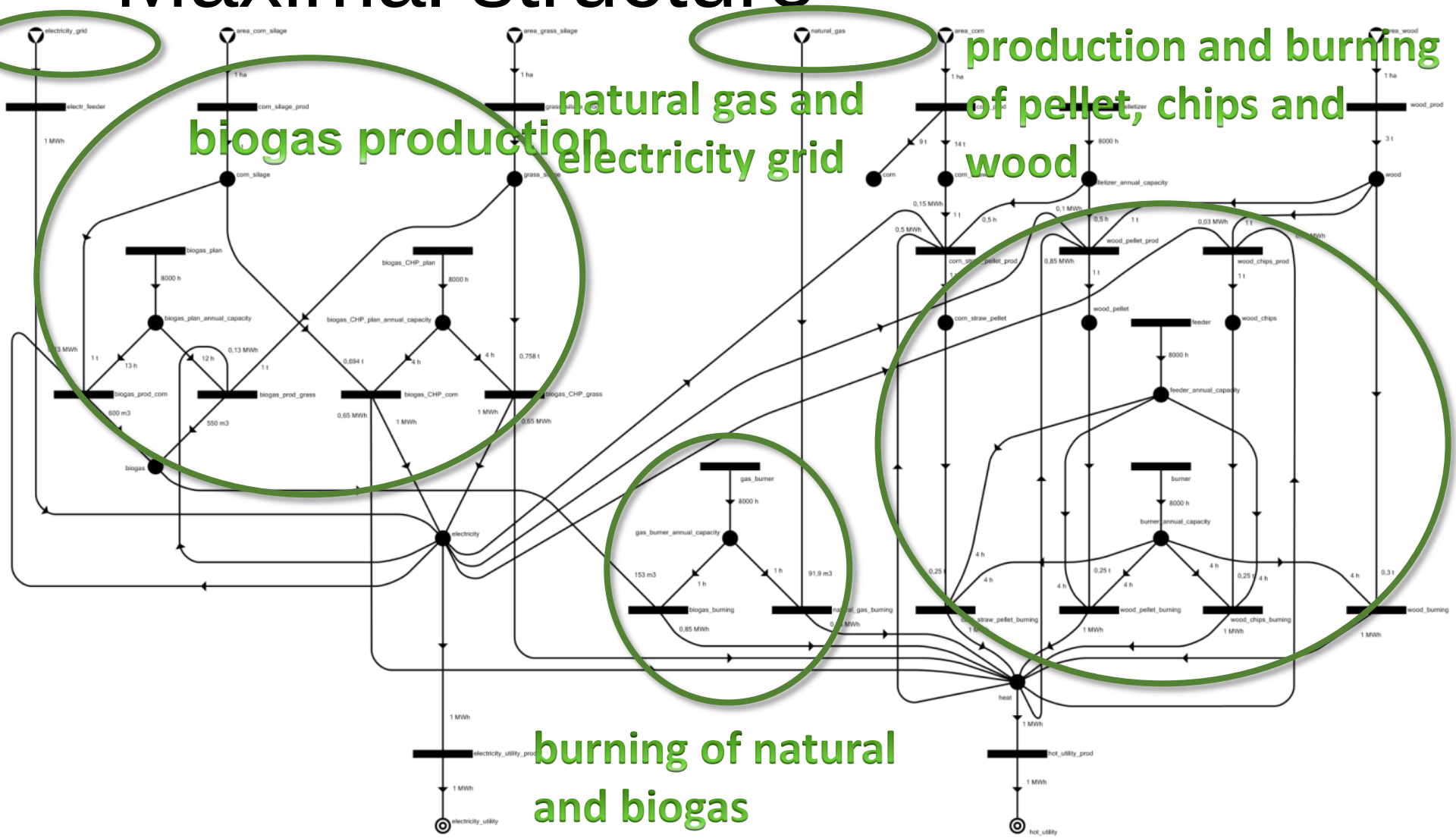
Software: PNS-Draw



Software: PNS-Studio



Maximal structure



Illustrative example

- An example is presented here to show the application of the P-graph framework while taking into account sustainability issues
- This graph represents the potential energy conversion technologies of a small region
  - conventional fossil energy sources are available like the natural gas and the electricity from the grid
  - this region has agricultural waste product in the form of grass and corn cobs, which can be used for biogas production

Illustrative example

- both the biogas and the natural gas can be fed to a furnace to produce heat
- there are available wood from which pellet and chips can be produced, and wood can be burned directly
- pellet also can be produced from corn cob

Data: Properties of the raw materials

Name	Cost		Max. flow	
electricity_grid	149	€/MWh		
natural_gas	0.5	€/m³		
area_corn			300	ha/yr
area_corn_silage			400	ha/yr
area_grass_silage			1 200	ha/yr
area_wood			600	ha/yr

Data: Properties of the products

Name	Min. flow	
hot_utility	5 000	MWh/yr
electricity_utility	2 000	MWh/yr

Data: The inputs and outputs of the operating units of the case study

Unit name	Input	Rate		Output	Rate		
electricity_feeder	electricity_grid	1	MWh	electricity	1	MWh	
biogas_plant				biogas_plant_c	8000	h	
biogas_prod_corn	biogas_plant_c	13	h	biogas	600	m^3	
	corn_silage	1	t				
	electricity	0.13	MWh				
biogas_prod_grass	biogas_plant_c	12	h	biogas	550	m^3	
	grass_silage	1	t				
	electricity	0.13	MWh				
corn_silage_prod	area_corn_silage	1	ha	corn_silage	12	t	
grass_silage_prod	area_grass_silage	1	ha	grass_silage	12	t	
biogas_CHP_plant				biogas_CHP_plant_c	8000	h	
biogas_CHP_corn	biogas_plan_c	4	h	electricity	1	MWh	
	corn_silage	0.694	t	heat	0.65	MWh	
biogas_CHP_grass	biogas_plant_c	4	h	electricity	1	MWh	
	grass_silage	0.758	t	heat	0.65	MWh	
electricity_utility_prod	electricity	1	MWh	electricity_utility	1	MWh	
gas_burner				gas_burner_c	8000	h	
biogas_burning	gas_burner_c	1	h	heat	0.85	MWh	
	biogas	153	m^3				
natural_gas_burning	gas_burner_c	1	h	heat	0.85	MWh	
	natural_gas	91.9	m^3				
corn_prod	area_corn	1	ha	corn	9	t	
				corn_straw	14	t	

Data: The inputs and outputs of the operating units of the case study

Unit name	Input	Rate		Output	Rate	
pelletizer				pelletizer_c	8000	h
corn_straw_pellet_prod	electricity	0.15	MWh	corn_straw_pellet	1	t
	heat	0.5	MWh			
	pelletizer_c	0.5	h			
	corn_straw	1	t			
wood_pellet_prod	electricity	0.1	MWh	wood_pellet	1	t
	heat	0.85	MWh			
	pelletizer_c	0.5	h			
	wood	1	t			
wood_chips_prod	electricity	0.03	MWh	wood_chips	1	t
	heat	0.48	MWh			
	wood	1	t			
wood_prod	area_wood	1	ha	wood	3	t
feeder				feeder_c	8000	h
burner				burner_c	8000	h
corn_straw_pellet_burning	corn_straw_pellet	0.25	t	heat	1	MWh
	feeder_c	4	h			
	burner_c	4	h			
wood_pellet_burning	wood_pellet	0.25	t	heat	1	MWh
	feeder_c	4	h			
	burner_c	4	h			
wood_chips_burning	wood_chips	0.25	t	heat	1	MWh
	feeder_c	4	h			
	burner_c	4	h			
wood_ burning	wood	0.3	t	heat	1	MWh
	burner_c	4	h			
hot_utility_prod	heat	1	MWh	heat_utility	1	MWh

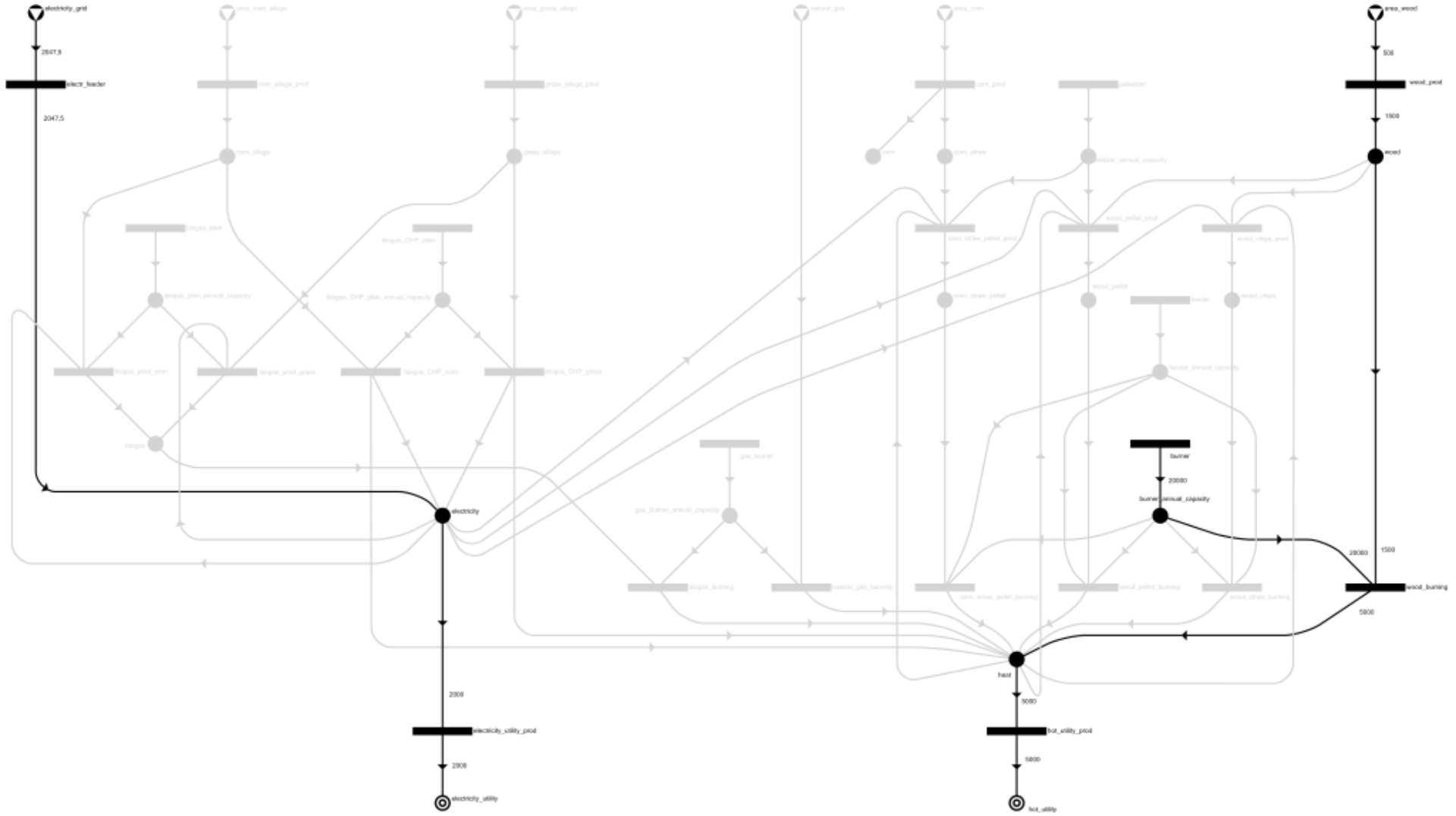
Data: The overall cost of the operating units of the case study

Unit name	Fixed part [€/yr]	Prop. part [€/yr]
electr_feeder	0	0
corn_silage_prod	0	960
grass_silage_prod	0	960
corn_prod	0	960
wood_prod	0	180
biogas_plant	35 000	49 286
biogas_CHP_plant	131 236	81 298
biogas_prod_corn	3 680	10
biogas_prod_grass	3 680	10
biogas_CHP_corn	9 822	4
biogas_CHP_grass	9 822	4
gas_burner	1 000	2 000
biogas_burning	0	0
natural_gas_burning	0	0

Unit name	Fixed part [€/yr]	Prop. part [€/yr]
burner	15 578	15 692
wood_burning	7 347	4
wood_chips_burning	7 347	3
wood_pellet_burning	7 347	3
corn_straw_pellet_burning	7 347	3
hot_utility_prod	0	0
wood_chips_prod	30 820	3
wood_pellet_prod	10 400	2
corn_straw_pellet_prod	10 400	2
electricity_utility_prod	0	0
pelletizer	30 000	185 000
feeder	100	0

Computational results

Optimal structure in terms of cost

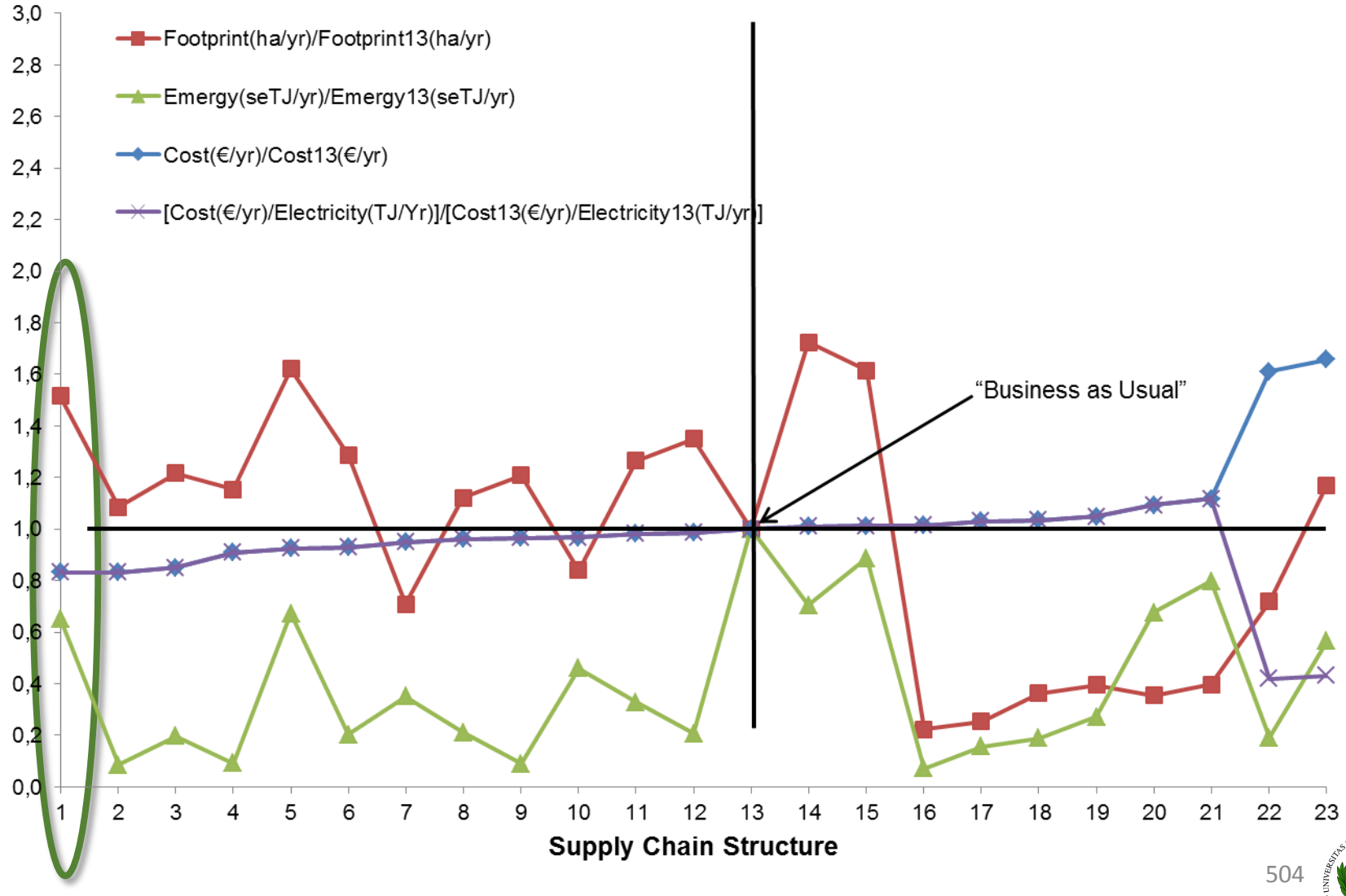


Computational time: < 1 s

Solution structures

Structure	Heat demand satisfied	Electricity demand satisfied	Cost [€/yr]	Ecological footprint	Emergy
#1	wood	grid	476,363	1046	1,764,910
#2	biogas CHP by corn silage, wood	biogas CHP by corn silage	476,433	749	234,594
#3	biogas CHP by grass silage, wood	biogas CHP by grass silage	486,852	840	532,100
#4	biogas CHP by corn silage, wood chips	biogas CHP by corn silage	521,283	796	245,744
...					
#13	natural gas	grid	572,956	690	2,706,600
...					

Results


Footprint(ha/yr)/Footprint13(ha/yr)
Emergy(seTJ/yr)/Emergy13(seTJ/yr)
Cost(€/yr)/Cost13(€/yr)
[Cost(€/yr)/Electricity(TJ/Yr)]/[Cost13(€/yr)/Electricity13(TJ/yr)]
"Business as Usual"
3,0
2,8
2,6
2,4
2,2
2,0
1,8
1,6
1,4
1,2
1,0
0,8
0,6
0,4
0,2
0,0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Supply Chain Structure

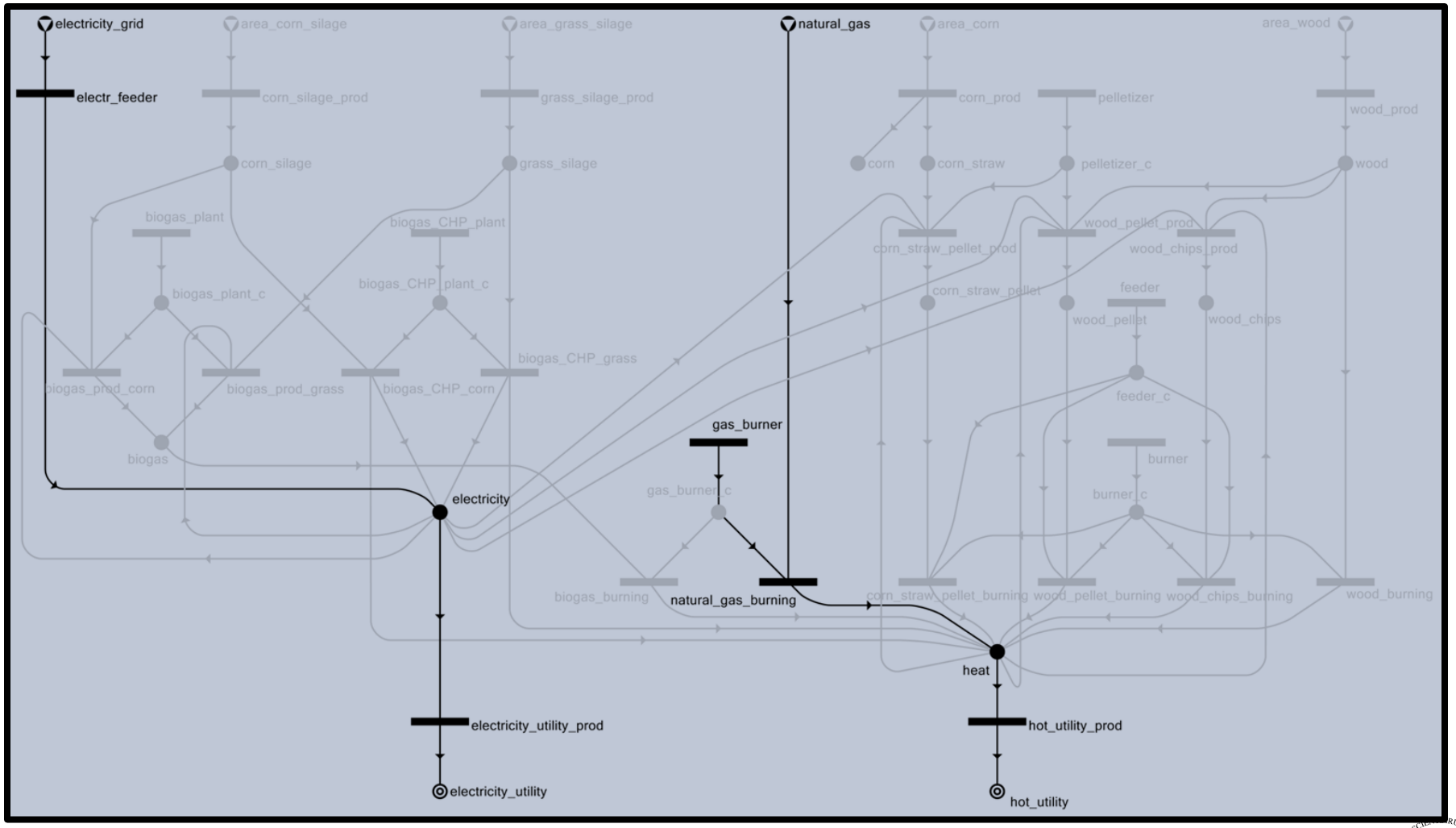
Raw Material & Energy Inputs

Solution structures	Raw materials					
	electricity_grid [TJ/yr]	natural_gas [m³/yr]	area_corn [ha/yr]	area_corn_silage [ha/yr]	area_grass_silage [ha/yr]	area_wood [ha/yr]
Structure1	7.37					500.00
Structure2				117.69		367.73
Structure3					128.54	367.73
Structure4				120.03		393.66
Structure5	7.57					539.13
Structure6					131.10	393.66
Structure7		**399,272.00**		**116.30**		
Structure8			72.96	126.77		
Structure9				124.78		380.69
Structure10		**399,272.00**			**127.02**	
Structure11			72.96		138.46	
Structure12					136.29	380.69
Structure13	**7.25**	**540,588.00**				
Structure14	7.99					529.10
Structure15	8.17		102.04			
Structure16				**214.45**		
Structure17				125.12	98.01	
Structure18				90.05	135.88	
Structure19					234.67	
Structure20	7.95			125.00		
Structure21	8.02				136.36	

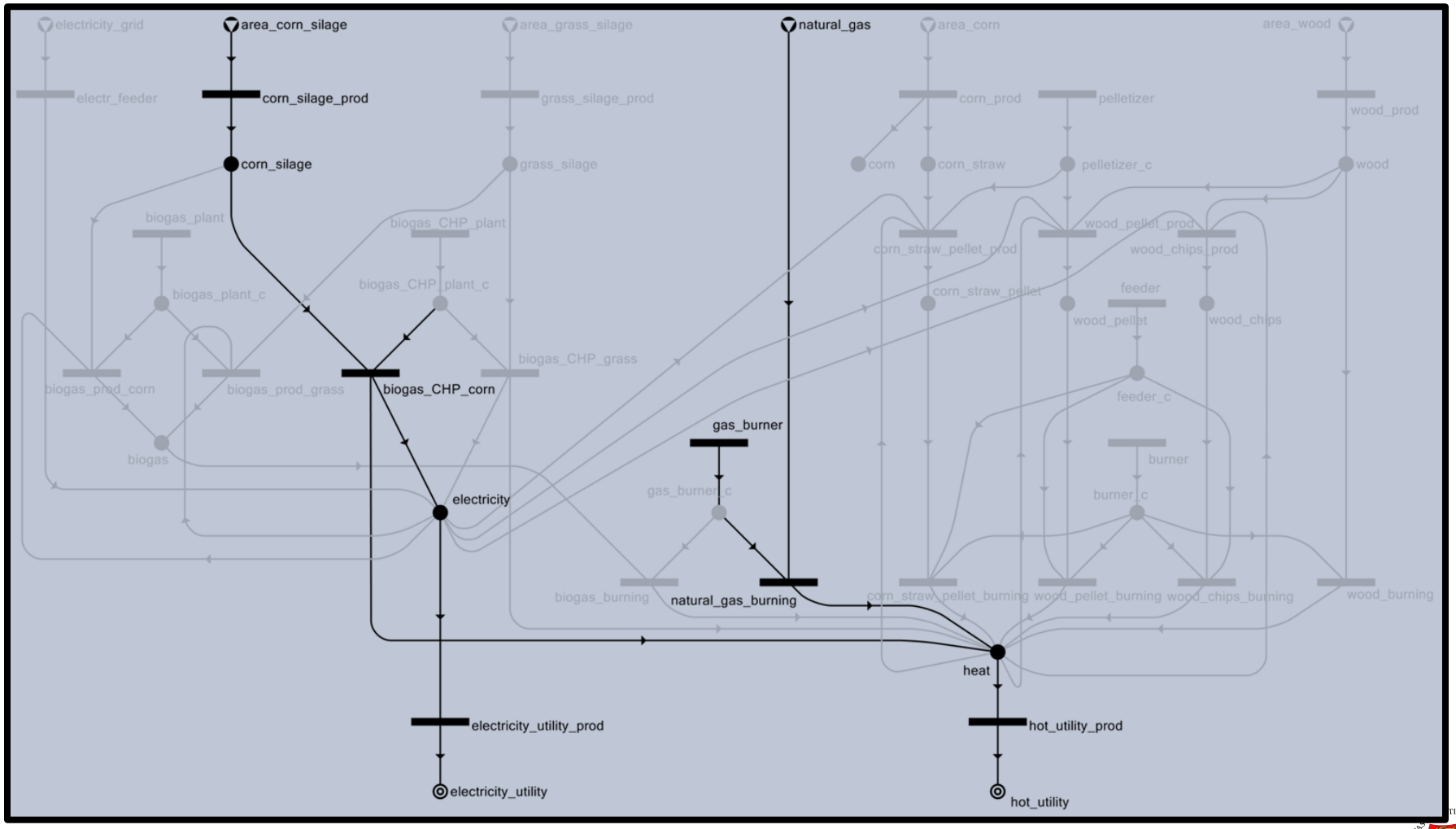
Computational results

- Multi-objective optimization usually has no clear winner
- Structure 1 has greater footprint than the base case (structure 13)
- Structure 2 is much better in terms of both ecological footprint and emergy and the cost is only slightly higher than the cost of Structure 1
- Structure 7 and 10 are better than the base case in emergy, ecological footprint, and cost as well
- Structure 16 is more expensive than the base case but there is a substantial drop in footprint here.
- Structure 22 and 23 produces more heat and electricity than required

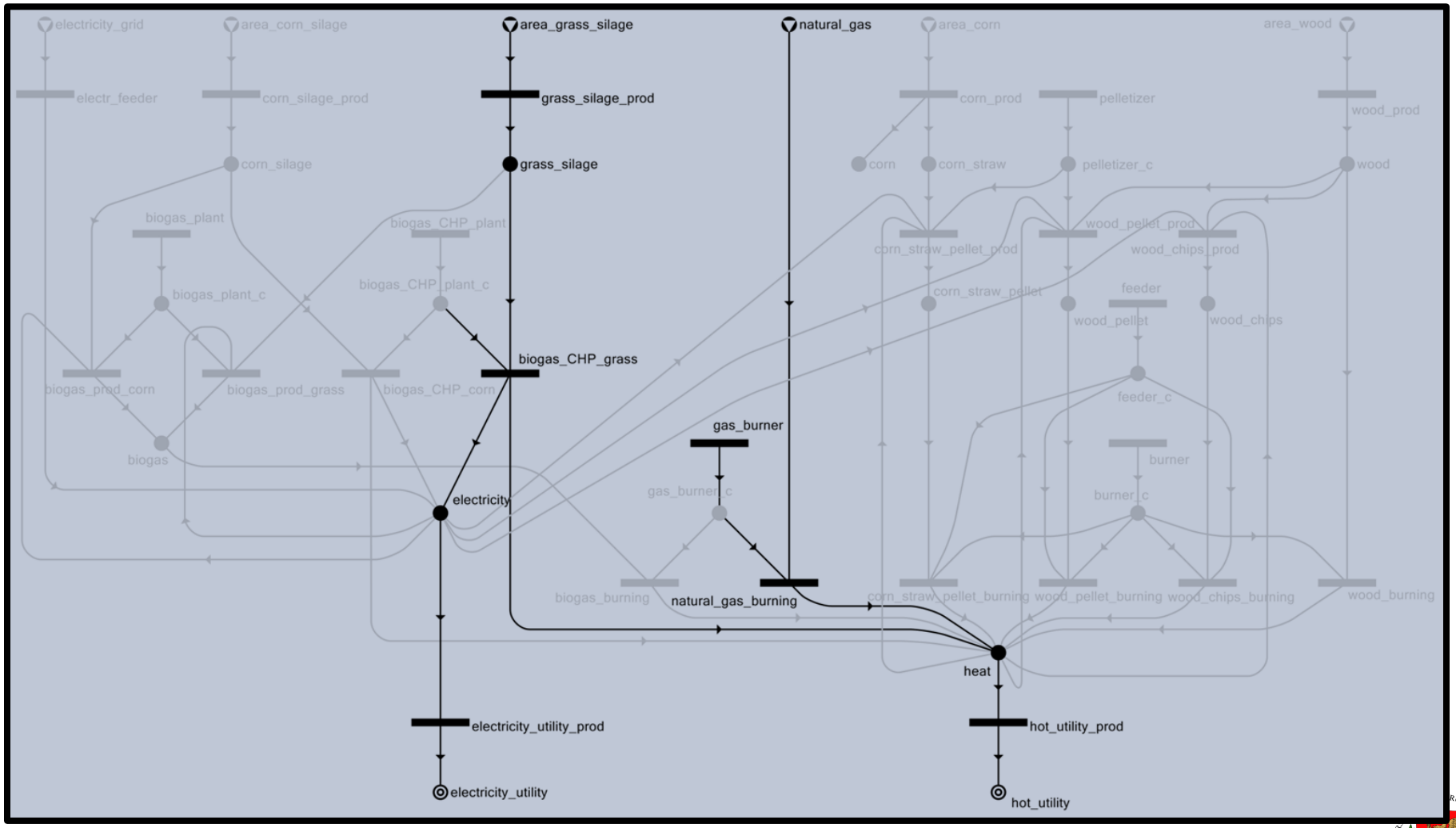
Structure #13: Natural Gas & Electricity from the “Grid”



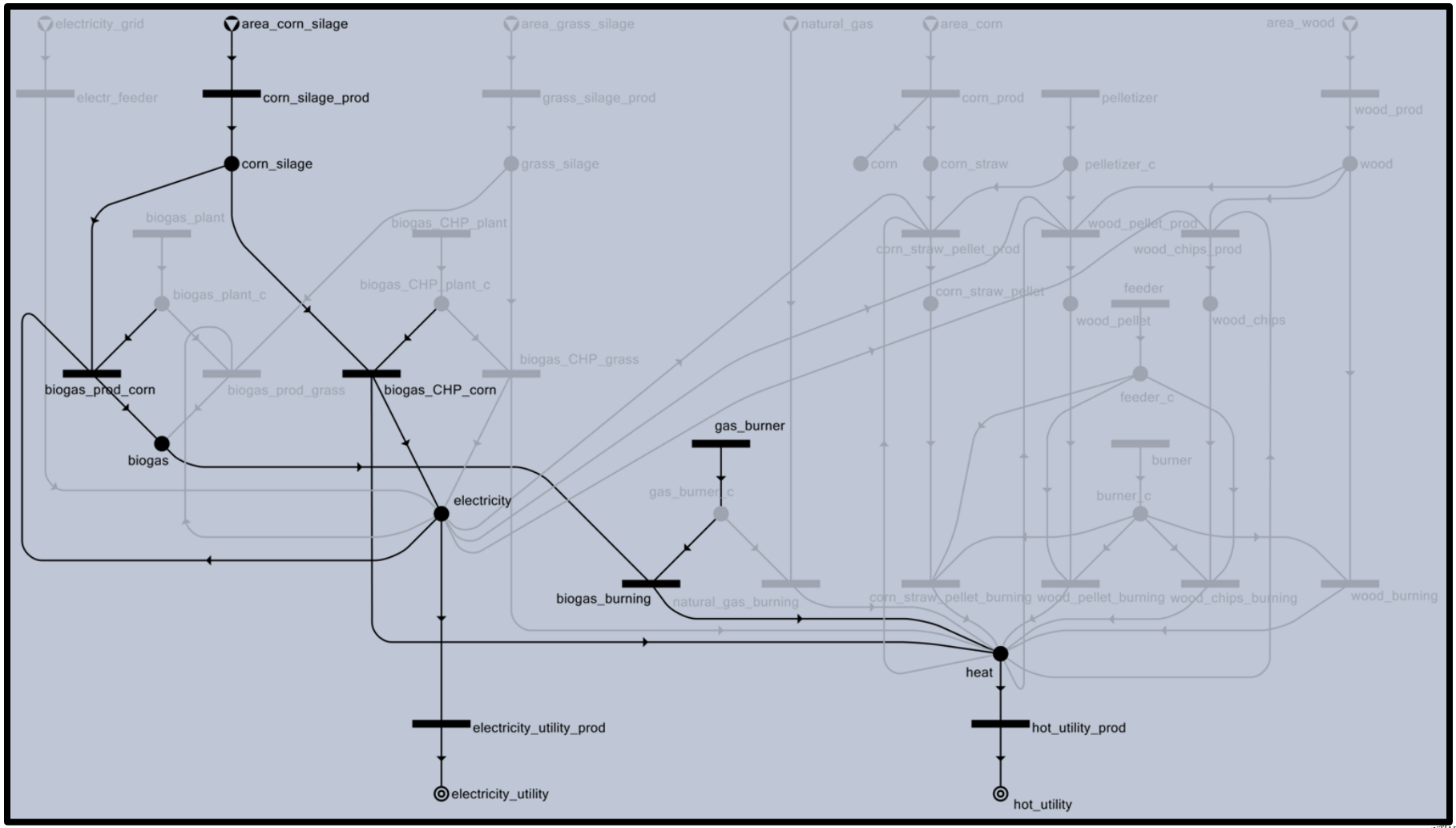
Structure #7: Corn Silage & Natural Gas



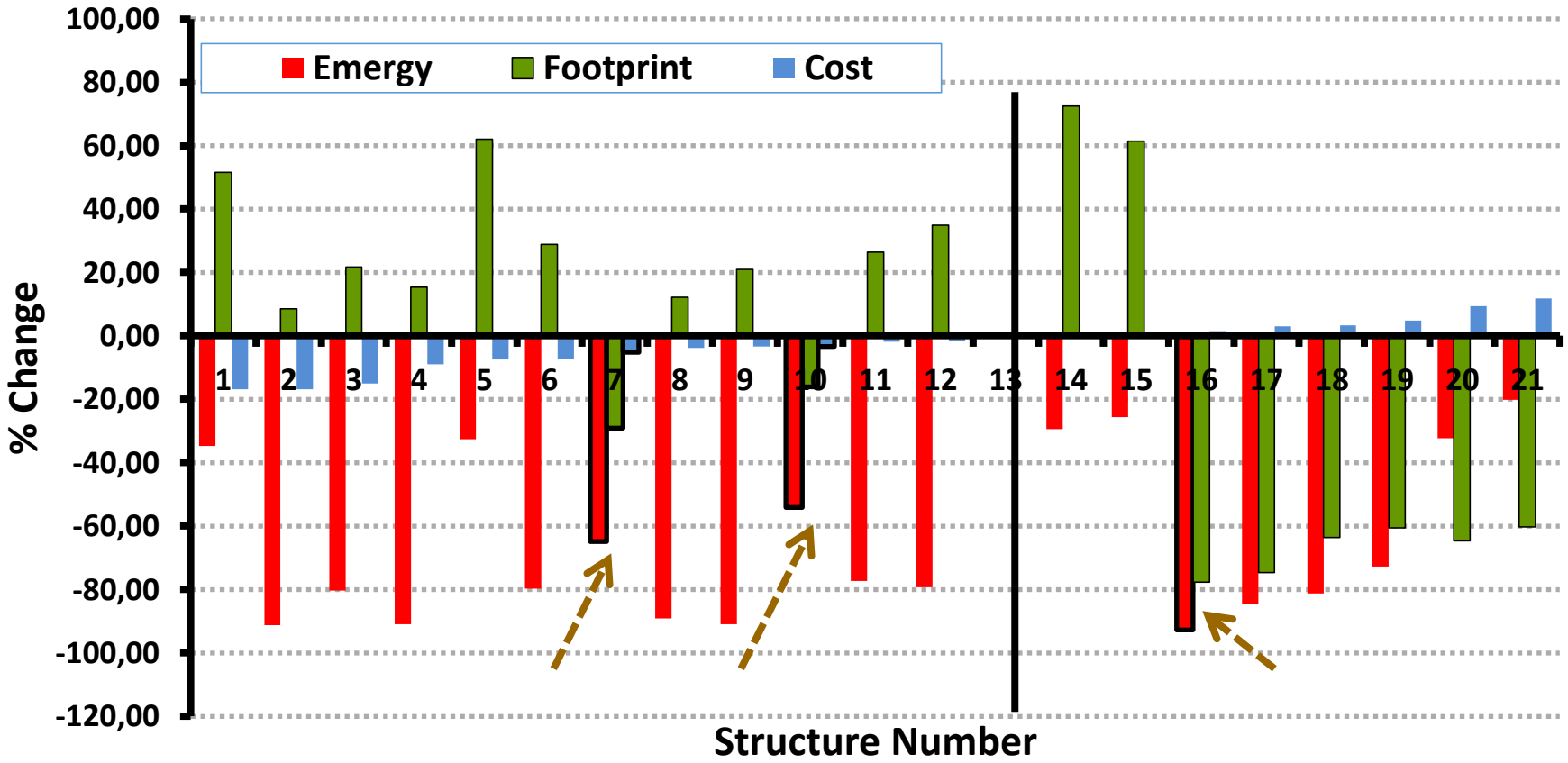
Structure #10: Grass Silage & Natural Gas



Structure #16: Corn Silage

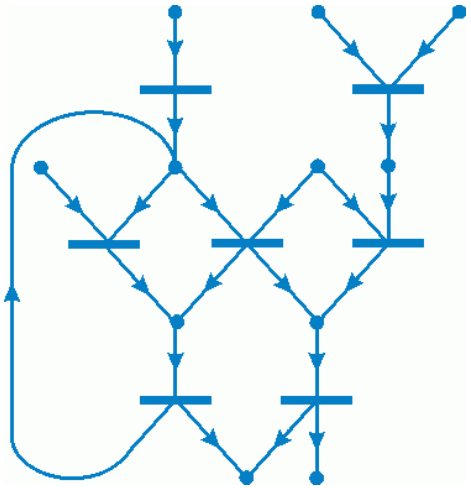


Supply Chain Structures



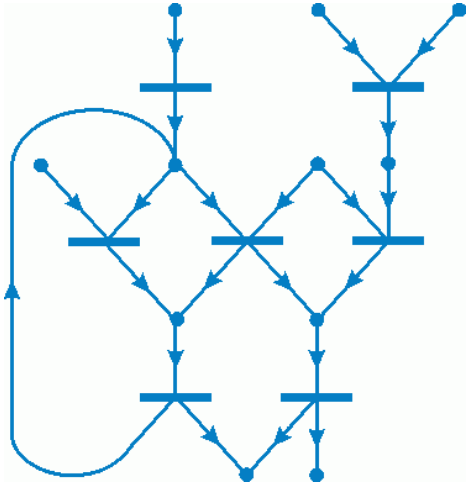
Summary

- Environmental protection
- The concept of sustainability
- P-graph framework
- The metrics of sustainability
- Multi-period operating unit
- Modeling of ecological footprint
- Illustrative example
- Computational results



Conclusions

- Sustainability is about adaptively managing the environment on an on-going basis so as to insure that the Earth can continue to support human existence for the indefinite future
- Carefully designed supply chains can be made both **cheaper and significantly more environmentally friendly** than current practice while meeting societal needs



ÚJ SZÉCHENYI TERV

A felsőfokú informatikai oktatás minőségének fejlesztése, modernizációja

TÁMOP-4.1.2.A/1-11/1-2011-0104

Főkedvezményezett:

Pannon Egyetem
8200 Veszprém
Egyetem u. 10.

Kedvezményezett:

Szegedi Tudományegyetem
6720 Szeged
Dugonics tér 13.

2014

MAGYARORSZÁG MEGÚJUL

A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

References

- Vance, L., H. Cabezas, I. Heckl, B. Bertok, and F. Friedler, Synthesis of Sustainable Energy Supply Chain by the P-graph Framework, Industrial & Engineering Chemistry Research, **52**(1), 266-274 (2013).
- Heckl, I., K. Kalauz, P. Kalocsai, and L. Halasz, Custom simulator for logistic networks in downstream, Clean Technology and Environmental Policy, **12**, 627-634 (2010).
- Varga, V., I. Heckl, F. Friedler, and L. T. Fan, PNS Solutions: A P-graph based programming framework for process-network synthesis problems, Chemical Engineering Transactions, **21**, 1387-1392 (2010).
- Heckl, I., F. Friedler, and L. T. Fan, Solution of separation network synthesis problems by the P-graph methodology, Computers & Chemical Engineering, **34**(5), 700-706 (2010).
- Heckl, I., P. Kalocsai, L. Halasz, and K. Kalauz, Event driven process simulation of pipeline networks, Chemical Engineering Transactions, 18, 737-742 (2009).
- Weber, C., I. Heckl, F. Friedler, F. Maréchal, and D. Favrat, Network synthesis for a district energy system: A step towards sustainability, Computer Aided Chemical Engineering, **21**, 1869-1874 (2006).

References

- Kettl, K. H., N. Niemetz, N. Sandor, M. Eder, I. Heckl, and M. Narodoslawsky, Regional Optimizer (RegiOpt) - Sustainable energy technology network solutions for regions, Computer Aided Chemical Engineering, 29, 1959-1963, (2011).
- K. Shahzad, R. Kollmann, S. Maier, M. Narodoslawsky, SPIonWEB – Ecological Process Evaluation with the Sustainable Process Index (SPI), Computer Aided Chemical Engineering, 33, 2014, 487-492
- M. Narodoslawsky, Chemical engineering in a sustainable economy, Chemical Engineering Research and Design, 91, 2013, 2021-2028
- G. Gwehenberger, M. Narodoslawsky, The ecological impact of the sugar sector- Aspects of the change of a key industrial sector in Europe, Computer Aided Chemical Engineering, 24, 2007, 1029-1034

Thank you for your attention!

Additional information: www.p-graph.com

Multiobjective PNS problems

Multiobjective optimization

- In multiobjective optimization more than one goals should be taken into account.
- Unfortunately it often happens that some solutions which have excellent performance in one objective have very week performance in the others.

Applications in case of PNS

Usually in process network synthesis we are looking for a cheapest solution, but other objectives can be important as well:

- One of the most important goals is to decrease the pollution of environment. Cleaner technologies are often more expensive therefore in these cases the objectives are very different.
- One can consider the execution time of the process as a second objective function. It is not sure that the faster execution is also a cheaper one.
- The stability of the production can be also a further objective.

The approaches to solve multiobjective problems

There are several methods to study multiobjective problems, we will overview the following ones

- Determining Pareto optimal or weakly efficient solutions
- Using aggregating functions to form a single-objective model
- Using Epsilon-Constraint method

Notations

- In general we suppose that we have k minimization functions denoted by $f_1, f_2, \ldots, f_k$. Note that it is not a restriction to consider minimization functions since taking the negative the maximization problem can be changed into a minimization one.
- The set of feasible solutions is denoted by S.
- In case of the PNS applications we will consider only two objective functions, the methods can be extended into the more general cases with some extra technical difficulties.

Notations in case of multiobjective PNS$_i$

- We mainly will consider the structural model, where an operating unit o_i has only fixed costs, denoted by cf_i and df_i. Then we have two objective functions: z_1 is the sum of the cf_i and z_2 is the sum of the df_i values of the selected operating units.
- We will also consider the more general fix charged linear cost model. Here the proportionality constants cp_i and dp_i are also assigned to the operating units, and z_1 is calculated by cf_i and cp_i and z_2 is calculated by df_i and dp_i.

Weakly efficient solutions

- We can say that a solution $x \in S$ is better than a solution $y \in S$ if it is better in each objective, which means that $f_i(x) < f_i(y)$ for each i.
- The solutions which are the best ones on this sense are called weakly efficient solutions.
- Formally we can say that a solution $x \in S$ is weakly efficient if there is no $y \in S$ such that $f_i(y) < f_i(x)$ is valid for each i.

Pareto optimal solutions

- On the other hand we can also say that a solution $x \in S$ is better than a solution $y \in S$ if it is better in at least one objective and not worse in the others.
- The solutions which are the best ones on this sense are called Pareto optimal solutions.
- Formally, a solution $x \in S$ is Pareto optimal if there are no $y \in S$ and j such that $f_i(y) \leq f_i(x)$ is valid for each i and $f_j(y) < f_j(x)$.

Aggregated objective function

- Let g be a k-variable monoton function which is called the aggregation function.
- Then we can form the single-objective optimization problem where we are looking for the $x \in S$ where $g(f_1(x), f_2(x),\ldots,f_k(x))$ is minimal.
- Usually g is a weighted sum with positive weigths but more general functions can also be used.

Theorem: *If g is a weighted sum with positive weigths then any optimal solution of the single objective aggregated problem is a Pareto optimal solution of the original one.*

Proof

We use an indirect proof:

- Let x be an optimal solution of the aggregated problem and suppose that it is not Pareto optimal.
- Then we have an $y \in S$ such that $f_i(y) \leq f_i(x)$ is valid for each i and there is a j with $f_j(y) < f_j(x)$.
- Since g is a weighted sum of f_i substituting x and y into g we obtain that $g(y)<g(x)$ which is a contradiction.

Epsilon-constrained method

- In the epsilon constrained method we have one distinguished objective function (suppose it is f_1) and the other objective functions are used to constrain the set of feasible solutions. In the model the bounds $C_2, C_3,\ldots,C_k$ are given and we consider only the solutions $x \in S$ which satisfy $f_i(x) \leq C_i$ for $i=2,\ldots,k$.

Theorem: *An optimal solution of any single-objective optimization problem received by the epsilon-constrained method is a weakly efficient solution of the original problem.*

Proof

- Let x be an optimal solution of the aggregated problem and suppose that it is not weakly efficient.
- This means that there exists $y \in S$ such that

$f_i(y) < f_i(x)$ is valid for each i.

- Then y is a feasible solution of the constrained problem, since by $f_i(y) < f_i(x) \leq C_i$ for $i=2,\ldots,k$.
- Then we obtain a contradiction by $f_1(y) < f_1(x)$.

Robust optimization

- In the optimizaton problems we often suppose that all costs are known exactly in advance. On the other hand, in real applications usually some uncertainty can change the data.
- In general, for most optimization model the problem of uncertainty is solved by stochastic optimization. On the other hand, in these cases we need some a priori information about the distribution of the data, which is usually not available in real applications.
- Another approach is a robust optimization, where the uncertainty is handled by deterministic worst case scenario. In these models we do not have the fixed values of the parameters we only know that they are in a given interval.

SSG based generation of Pareto optimal PNS solutions in structural PNS problems

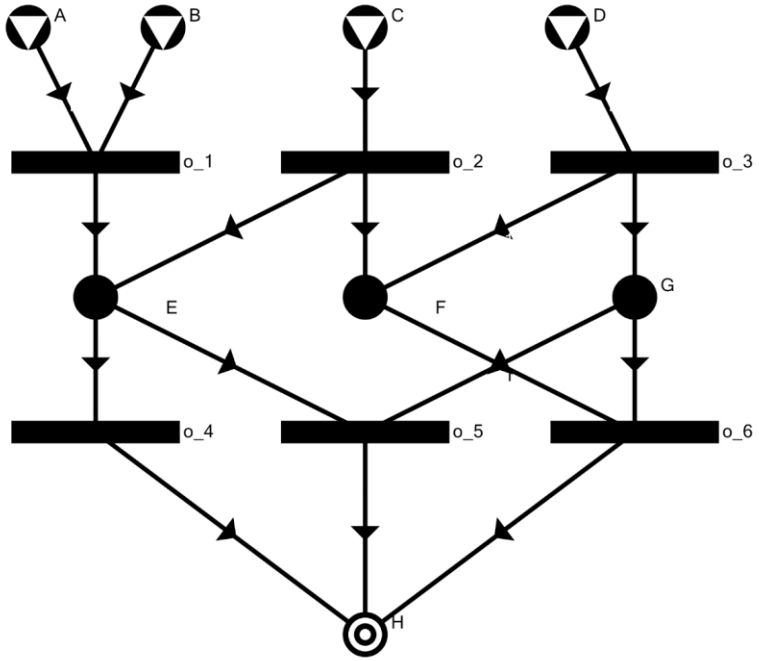
- In the structural model the cost of a solution depends only on the operating units contained in it, therefore the SSG algorithm which lists all of the feasible solutions can be extended to determine the Pareto optimal ones.
- We have to use a candidate set *J* and in each step when a new solution is found we upgrade this set.
- If the solution is worse than some elements of *J*, then *J* is not changed.
- Otherwise the actual solution is put into *J* and those earlier elements of *J* which are worse are deleted.

Example

- Consider the problem shown in the next figure where 6 operating units o_1, $o_2,\ldots,o_6$ are defined.
- The cost are given as follows:

	1	2	3	4	5	6
cf	3	1	2	9	6	3
df	2	3	2	8	2	6

The maximal structure of the example



The Pareto optimal solutions received by extended SSG

- The SSG algorithm lists all of the solutions. The first is $x_1=\{o_1, o_4\}$ with $z_1(x_1)=12$ and $z_2(x_1)=10$ and we put it into the candidate list J.
- The second solution is $x_2=\{o_2, o_4\}$ with $z_1(x_2)=10$ and $z_2(x_2)=11$, it is not worse than x_1 thus we put it into the candidate list J.
- The third solution is $x_3=\{o_1 o_2, o_4\}$ with $z_1(x_3)=13$ and $z_2(x_3)=14$, it is worse than x_1 thus we do not put it into the candidate list J.
- The next solution is $x_4=\{o_1 o_3, o_5\}$ with $z_1(x_4)=11$ and $z_2(x_4)=6$, it is better than x_1 and not worse than x_2 thus we put it into the candidate list J, and x_1 is deleted.

The Pareto optimal solutions received by extended SSG

- The next solution is $x_5=\{o_2, o_3, o_5\}$ with $z_1(x_5)=9$ and $z_2(x_5)=7$, it is better than x_2 and not worse than x_4 thus we put it into the candidate list J, and x_2 is deleted.
- The next solution is $x_6=\{o_1, o_2, o_3, o_5\}$ with $z_1(x_6)=12$ and $z_2(x_6)=9$, it is worse than x_5 thus we do not put it into the candidate list J.
- The next solution is $x_7=\{o_3, o_6\}$ with $z_1(x_5)=5$ and $z_2(x_5)=8$, which is neither worse nor better than x_4 and x_5 thus we put it into J.
- All of the remaining 13 solutions are worse than one of the elements of J, thus the set of the Pareto optimal solutions is $\{x_4, x_5, x_7\}$

Branch and bound based generation of Pareto optimal PNS solutions in structural PNS problems

- Usually a PNS problem has a lot of feasible solution and only a few of them are Pareto optimal. Therefore an algorithm which does not generate all of the feasible solutions can be more effective.
- We can also extend the Branch and Bound based ABB algorithm to generate Pareto optimal solutions.
- In this case we can exclude the sets of the feasible solutions where we know by the bounding functions that all solutions are worse then one of the elements in *J*.

Extension to the fix charged linear cost model

- In the structural model we have only a finite number of feasible solutions. In the more general version the material flows are alos considered, thus we have an infinite set of solutions.
- On the other hand we again have only finite number of structures which can be generated by SSG. Thus the problem is reduced to find the Pareto optimal solutions in case of fixed structures.
- This means that we have to solve a multiobjective linear programming problem which is widely studied.

Linear aggregated cost functions

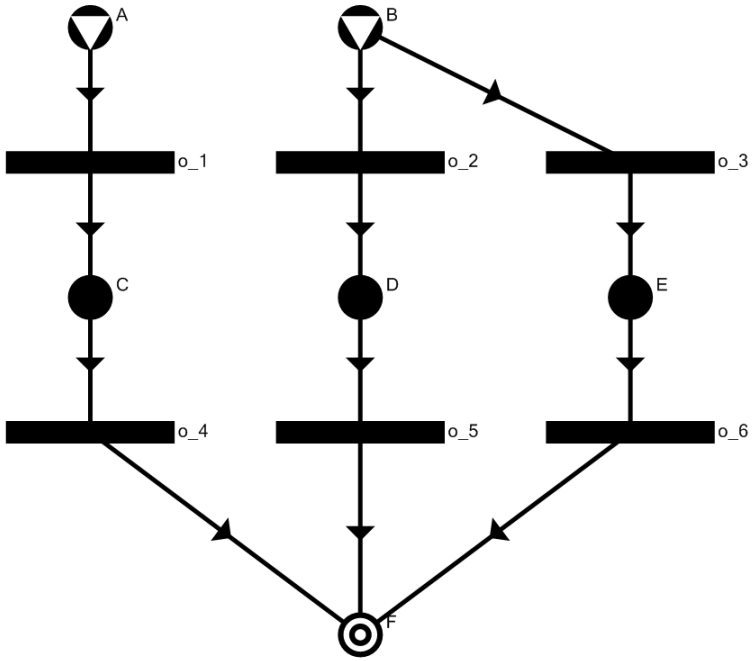
- If there is a linear aggregation function then we can reduce the problem to the solution of a PNS problem.
- If the the aggregated function $z(x)= r_1z_1(x)+ r_2z_2(x)$ then we can define the following PNS problem with the same set of operating units.
- For operating unit o_i the fix cost will be $ef_i= r_1cf_i+ r_2df_i$ and the propotionality cost will be $ep_i= r_1cp_i+ r_2dp_i$.
- Then the cost of a feasible solution of this new PNS problem will be the aggregated cost of the original problem.

Example

- Consider the problem shown in the next figure where 6 operating units o_1, o_2,…,o_k are defined.
- The cost are given as follows:

	1	2	3	4	5	6
cf	6	3	1	9	5	3
df	2	4	7	4	5	7

The maximal structure of the example



The aggregated optimal solutions

- We have three solutions such that any further solution contains some of them: $x_1=\{o_1, o_4\}$ with $z_1(x_1)=15$ and $z_2(x_1)=6$, $x_2=\{o_2, o_5\}$ with $z_1(x_2)=8$ and $z_2(x_2)=9$, $x_3=\{o_3, o_6\}$ with $z_1(x_3)=4$ and $z_2(x_3)=14$.
- If we consider $z_1$then x_3 is optimal.
- If we consider z_2 then x_1 is optimal.
- If we consider the aggregated objective $z=z_1+z_2$, then x_2 is optimal.
- We note that this also shows that all of these solutions are Pareto optimal.

Nonlinear aggregating functions

- If we use more difficult aggregation functions then the problem can not be reduced into a single obective PNS problem.
- In these cases we can extend the ABB algorithm into a version which can solve the problem. The set of the feasible solutions is independent on the objective function therefore we only have to define new bounding functions.
- The simplest way to define a new bound is to use the bounds given on the objectives z_1 and z_2. If at some point we have bounds L_1 and L_2 on a set of feasible solutions, then $g(L_1,L_2)$ will bound the aggregated function on this set.

Nonlinear aggregating functions(2)

- On the other hand handling the objectives separately can yield weak bounds as the following example shows.
- Suppose we consider the set which contains three solutions: x_1 with $z_1(x_1)=12$ and $z_2(x_1)=40$, x_2 with $z_1(x_2)=25$ and $z_2(x_2)=25$, x_3 with $z_1(x_3)=35$ and $z_2(x_3)=15$. And let $g(x,y)=x^2+y^2$.
- Then the best bound on z_1 is 12, the best bound on z_2 is 15, thus using the bounds separately we cannot obtain better bound than 369 on the aggregated function. On the other hand the minimal value of the aggregated objective is 1250.

Epsilon-constrained method

- In this version the set of the feasible solutions and also the optimal solution are changing as the bounds are changed. Consider the example used in the aggregated problem.
- If $C_2=14$, then x_1,x_2,x_3 are all feasible thus x_3 is the optimal solution with $z_1(x_3)=4$.
- If we use $C_2=9$, then x_3 is excluded thus x_2 will be the optimal solution with $z_1(x_2)=8$.
- Finally, if we use $C_2=6$, then x_2 is also excluded thus x_1 will be the optimal solution with $z_1(x_1)=15$.

Branch and bound based approach

- In this model the objective is z_1 but the set of feasible solutions is changed, therefore we have to extend to ABB algorithm to handle this restricted sets of feasible solutions.
- One basic idea is to use the bounding function on z_2 to exclude some subset of solutions.
- If for a set of feasible solutions using the bounding function for z_2 we obtain that z_2 is at greater than C_2 for all of these solutions then we can exclude this set.

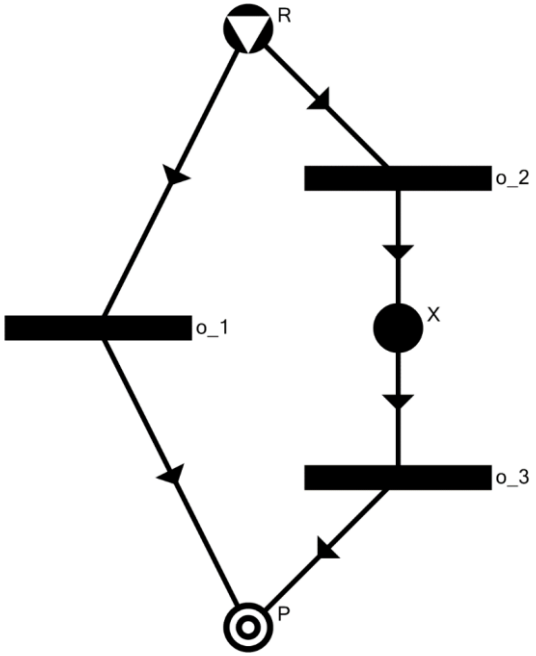
Robust PNS model

- In the robust model each operating unit o_i has an extended cost $c(o_i)+ e(o_i)$. We will call $c(o_i)$ the nominal cost and $e(o_i)$ the extra cost of the operating unit.
- We have an a priori bound b, which means that b operating units can have the extended cost and the others have the nominal cost.
- We are interested in the worst case, therefore, if we consider a feasible solution of the problem in the robust version its cost will be the sum of the nominal costs of the operating units plus the sum of the b largest extra costs.

Example

- Consider the PNS problem of the next figure where there are three operating units: o_1 with $c(o_1)=5$, $e(o_1)=2$, o_2 with $c(o_2)=2$, $e(o_2)=2$ and o_3 with $c(o_3)=2$, $e(o_3)=2$.
- If we consider the standard problem then the optimal solution contains o_2 and o_3 and the optimal cost is 4.
- If we consider the robust version with b=1, then the optimal solution still contains o_2 and o_3 and the optimal cost is 6.
- But if we consider the robust version with b=2, then the optimal solution contains o_1 and and the optimal cost is 7.

The maximal structure of the example



Branch and bound based algorithm

- The set of feasible solution is the same as in the standard PNS problem therefore we have to extend the bounding function.
- The simplest function contains the total nominal cost of the selected operating units plus the sum of the b greatest extra costs among them.
- We receive a more difficult function if we increase this simplest one by the shortest path from the raw materials into the selected operating units, where the cost of a path is the sum of the nominal costs of the operating units contained in it.

Heuristic algorithm for the structural model

- Both the SSG and ABB algorithms have exponential running time in the worst case, therefore some huge problems migh not be solved by them.
- In these cases heuristic algorithms which produce a good feasible solution in a short time can be useful.
- Moreover, these algorithms can be also used to accelerate the branch and bound algorithms giving a good starting solution which increases the efficiency of excluding subsets.

ASUM and AMAX heuristics(1)

- These greedy type algorithm use estimations on the cummulated costs of the materials (MA) and operating units (OP).
- The algorithms builds a solution step by step adding every time an operating unit to the solution.
- For each operating unit it calculates the sum of the estimated production cost of the input materials and the estimated cost of the operating unit. This sum is divided by the number of desired materials produced by the operating unit. The algorithm chooses the operating unit where this ratio is minimal.

ASUM and AMAX heuristics(2)

- At the beginning the set of desired material is the set of the desired products. Later in each step the input materials of the selected operating unit is put into this set and its output materials are deleted.
- The difference in the heuristics is in calculating the estimated costs of the input materials. ASUM takes the sum of the *MA* values, AMAX consider the maximum of the *MA* values.
- Checking the axioms S1 to S5 one can prove easily that the heuristics give a feasible solution for every function *MA* and *OP.*

OP function

- In the estimation of the cost of the operating unit we have to face with the problem of robustness, we do not know whether the nominal or the extended cost will be used in the solution. We can use the following solutions.
- Weighted cost: In this case we use some weighted average of the two costs thus $OP(o_i)=\alpha c(o_i)+(1-\alpha)(c(o_i)+ e(o_i))$ for some $0\leq\alpha\leq1$.
- Worst case cost: In this case we use the extended cost unless we already selected *b* operating units with at least as big extra cost as the actual unit has. In the latter case we use the nominal cost.
- Hibrid cost: we use the weighted cost in the first case instead of the worst case.

MA function

- The estimated cost of the materials depends on the cost of the operating units, thus we will use here the weighted cost of the operating units.
- We use the function which was used to calculate lower bound in some branch and bound algorithm changing the cost of the operating unit into the estimation.
- The general definition of this cost function is very difficult therefore we will only define it for cycle free PNS problems below.

MA function in cycle free P-graphs

- We will use two sets of materials *I* denotes the materials with the given *MA* values and *J* denotes the complement set where we have to calculate the value of MA. At the beginning *I* contains the raw materials with *MA(m)=0*, and later in each step one element is moved from J to I.
- We always choose such a material *m* from *J* which is only produced by operating units having input materials in *I.*
- We calculate a production cost *c* for each such operating unit producing *m* as follows.

MA function in cycle free P-graphs

- Calculate the sum of the maximum of the MA values of the input materials and the weighted OP cost of the operating unit. (Note that by the definition of m we know that MA is known for all inut materials.)
- Let *MA(m)* be the minimum of the production costs calculated above for the operating units producing *m*. We move *m* from *J* to *I.*
- The procedure ends when *J* becomes empty which means that *MA* is calculated for all materials.

Robust extension of the fix charged linear cost model

- We can define in two ways the robust extension of the fix charged linear cost model.
- If we use robustness only in the fix costs then we can extend the ABB algorithm in the same way as in the structural model changing only the bounding function.
- If we define robustness also in the proportionality constants, then we have to solve robust linear programming problems in the bounding function of the branch and bound algorithm.

Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: axioms and theorems. Chem. Eng. Sci. 1992, 47, 1973.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Combinatorial algorithms for process synthesis. Comput. Chem. Eng. 1992, 16, S313.
- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- http://www.p-graph.com

Solving standard optimization problems with the P-graph framework

Goals

- Introduction of modelling techniques of the P-graph framework for problems featuring properties different from those of the original chemical process planning area.
- The standard optimization problems do appear seldomly in their original form in real life. In most of the cases, practical problems include other parameters and constraints, that are difficult to implement in the dedicated algorithms. In contrast, the models presented here can be extended more easily for additional problem paramaters.

Considered problems

- Minimal spanning tree
- Shortest path
- Maximal flow
- Transportation problem

Finding the minimal spanning tree

- Given is a weighted graph $G=(V,\mathcal{E},w)$, where
  - V is the set of vertices
  - $\mathcal{E}$ is the set of edges (an edge is a set of two vertices)
  - $W : \mathcal{E} \rightarrow \mathbb{R}$ is a weight function for the edges
- The objective is to find a subgraph $G' \subseteq G$ such that
  - G' is a tree
  - The sum of the weights of the edges in G' is minimal
- The problem can be solved to optimality efficiently by using the algorithms of Prim or Kruskal

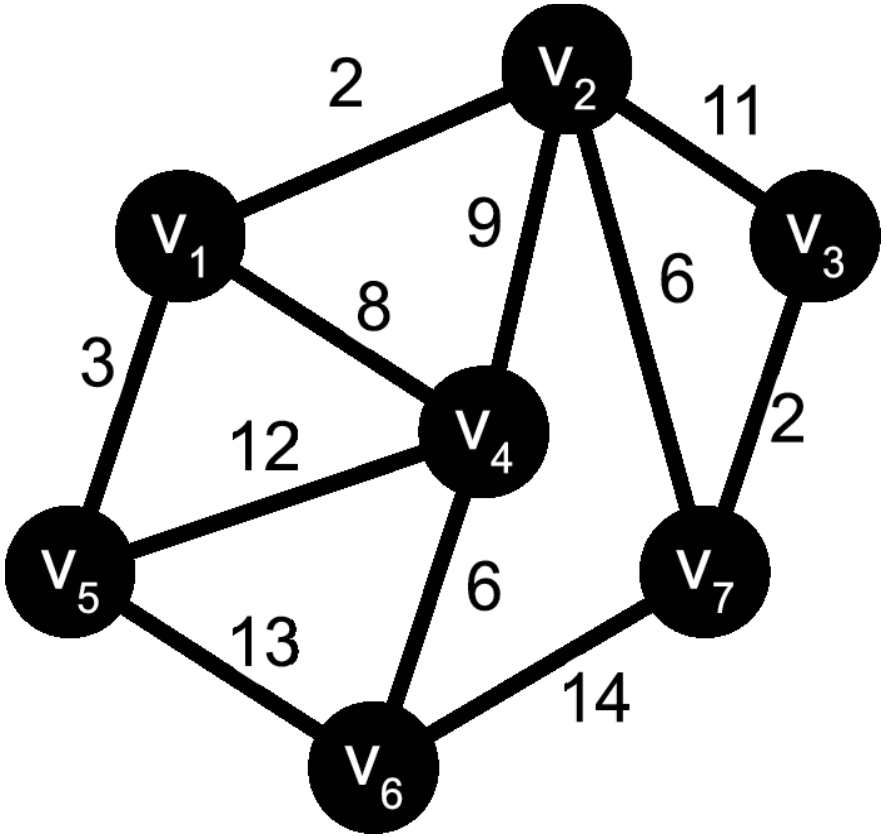
Finding the minimal spanning tree with the P-graph framework

- Modelling
  - Materials correspond to vertices of the original graph
    - To a dedicated vertex, *v** a raw material is assigned. All the other materials are of product type.
    - The maximal amount of the used *v** is $|V|-1$,
    - For all the other materials (products), the required quantity is 1.
    - Prices, costs are not assigned to any of the materials.
  - Operating units correspond to the edges of the graph
    - For all *{v,v'}* edge, two operating units are assigned: ({v},{v'}) and ({v'},{v})
    - The fix cost is given by the weight of the edge: *w({v,v'})*
    - Propotional costs and capacity limits are not introduced.

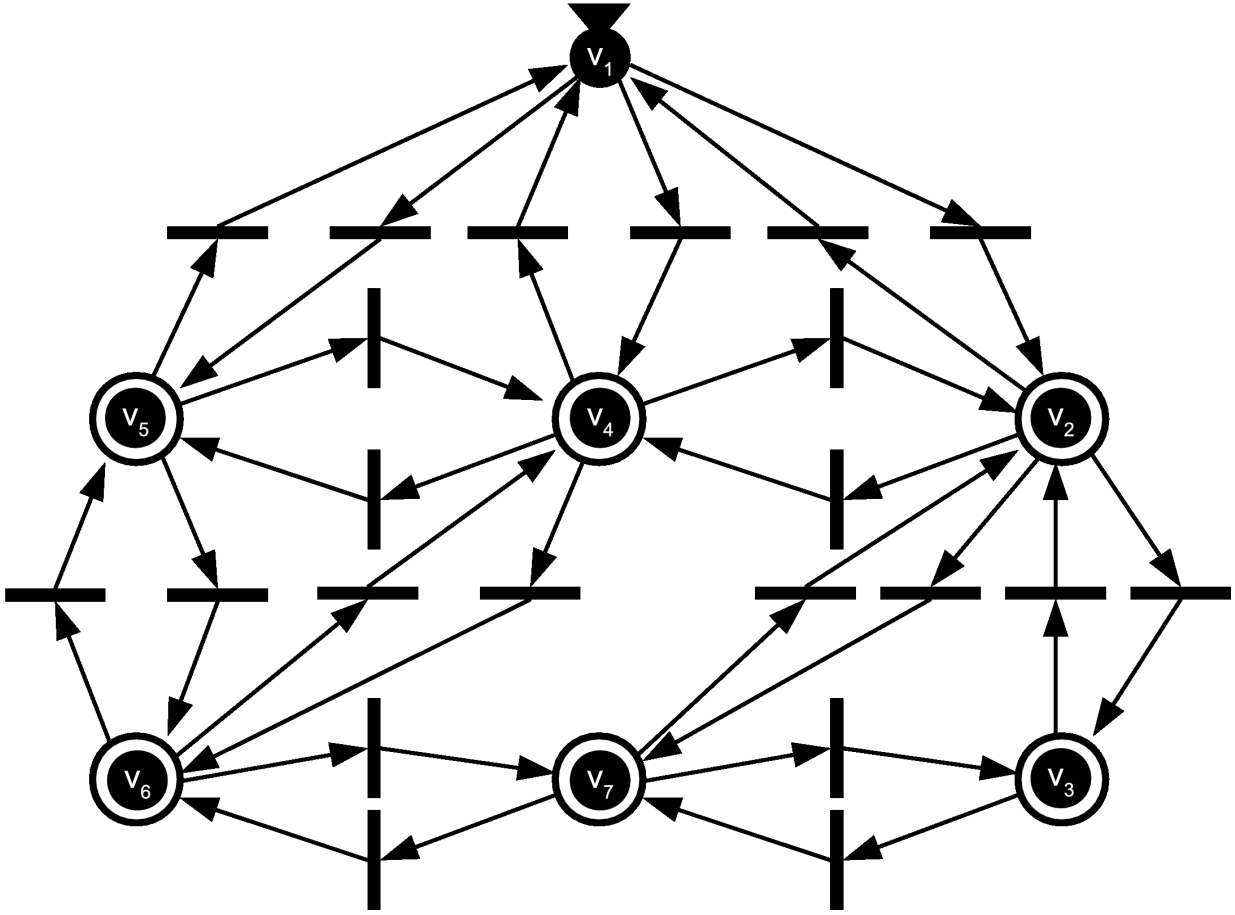
Finding the minimal spanning tree with the P-graph framework, cont'd

M	$=$	V
R	$=$	$\{v^*\}$, where v^* is arbitrary chosen from V
I	$=$	$\emptyset$
P	$=$	$V \setminus \{v^*\}$
O	$=$	$\{(\{v_1\},\{v_2\}) \mid \{v_1,v_2\} \in \mathcal{E}\}$
min_m	$=$	$\begin{cases} \lvert V\rvert - 1 & m = v^* \\ 1 & \text{otherwise} \end{cases}$
max_m	$=$	$\begin{cases} \lvert V\rvert - 1 & m = v^* \\ 1 & \text{otherwise} \end{cases}$
cap_o	$=$	∞ for all $o \in O$
fix_o	$=$	$w(\{v_1,v_2\})$, where $o = (\{v_1\},\{v_2\})$
$prop_o$	$=$	0 for all $o \in O$
$price_m$	$=$	0 for all $m \in M$

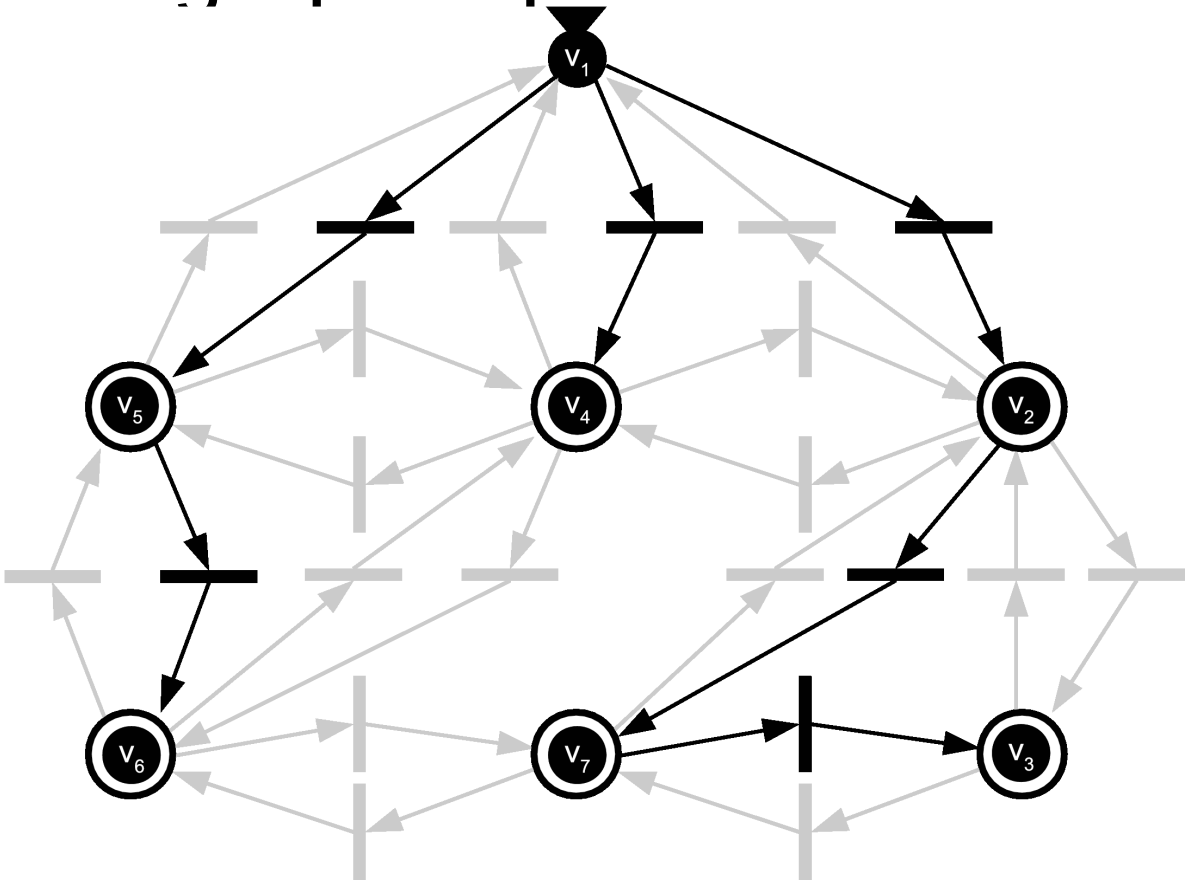
Finding the minimal spanning tree Example



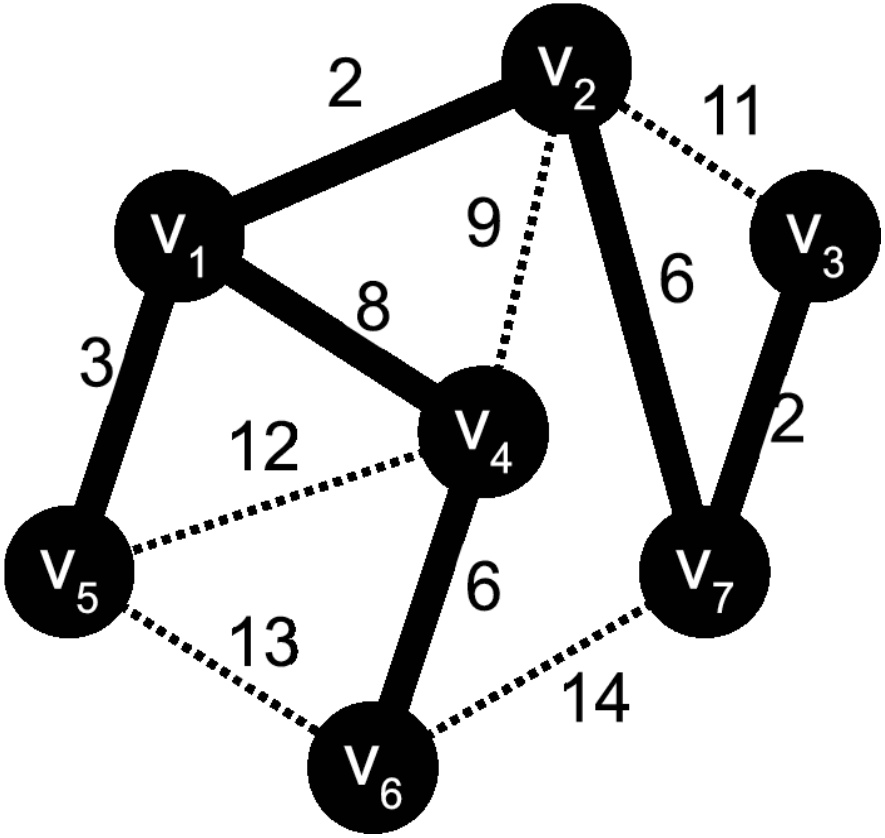
Finding the minimal spanning tree
P-graph model for the example



Finding the minimal spanning tree Solution given by the ABB algorithm for the P-graph representation



Finding the minimal spanning tree Solution of the example based on the optimal PNS structure



Finding the shortest path

- Given is a weighted directed graph D=*(V,A,w)*, a source vertex *s*, and a destination vertex *d*, where
  - *V* is the set of vertices
  - $A \subseteq V \times V$ is the set of arcs
  - $W : A \rightarrow \mathbb{R}$ is a weight function for the arcs
- The objective is to find a path from *s* to *d* with minimal weight sum
- The problem can be solved to optimality efficiently by using the algorithm of Dijkstra.

Finding the shortest path with the P-graph framework

- Modelling
  - Materials correspond to the vertices of the graph
    - The source vertex s is assigned with a raw material
    - The destination vertex *d* belongs to a product
    - All the other vertices are represented by an intermediate
    - The consumed amount from s, and the required quantity from *d* is 1, while it is forbidden to remain any amount of the intermediates
    - Prices, costs are not introduced for the materials.
  - Operating units correspond to the arcs
    - For each arc *(v,v'),* an operating unit ({v},{v'}) is introduced
    - The fix cost is the weight of the arc, *w((v,v'))*
    - Proportional cost are not introduced, and the capacity limit is 1 for all of the units.

Finding the shortest path with the P-graph framework cont'd

M	$=$	V
R	$=$	$\{s\}$
I	$=$	$V \setminus \{s, d\}$
P	$=$	$\{d\}$
O	$=$	$\{(\{v_1\}, \{v_2\}) \mid \{v_1, v_2\} \in A\}$
min_m	$=$	$\begin{cases} 1 & m = s, d \\ 0 & \text{otherwise} \end{cases}$
max_m	$=$	$\begin{cases} 1 & m = s, d \\ 0 & \text{otherwise} \end{cases}$
cap_o	$=$	1 for all $o \in O$
fix_o	$=$	$w((v_1, v_2))$, where $o = (\{v_1\}, \{v_2\})$
$prop_o$	$=$	0 for all $o \in O$
$price_m$	$=$	0 for all $m \in M$-

Finding maximal flow

- Given is a weighted directed graph D=*(V,A,w)*, a source vertex *s*, and a destination vertex *d*, where
  - *V* is the set of vertices
  - $A \subseteq V \times V$ is the set of arcs
  - $W : A \rightarrow \mathbb{R}$ is a weight function for the arcs.
- The objective is to find a wight function *w'* such that
  - *W'(a) ≤ w(a)* holds for all of the arcs
  - The Kirchoff junction law is satisfied for all vertices except *s* and *d*
  - The overall weight of the arcs leading from *s* is maximal
- The problem can be solved to optimality efficiently by the algorithms of Ford & Fulkerson

Finding the maximal flow with the P-graph framework

- Modelling
  - Materials correspond to the vertices of the graph
    - The source vertex s is assigned with a raw material
    - The destination vertex *d* belongs to a product
    - All the other vertices are represented by an intermediate
    - Minimal and maximal limits are not introduced for the materials
    - The price of the product is 1, the other materials does not have a price, cost or penalty assigned.
  - Operating units correspond to the arcs
    - For all (*v,v'*) arc an operating unit ({v},{v'}) is assigned.
    - The capacity limit is the weight of the arc, i. e., *w((v,v'))*
    - *Proportional and fixed costs are not introduced.*

Finding the maximal flow with the P-graph framework cont'd

M	$=$	V
R	$=$	$\{s\}$
I	$=$	$V \setminus \{s, d\}$
P	$=$	$\{d\}$
O	$=$	$\{(\{v_1\}, \{v_2\}) \mid \{v_1, v_2\} \in A\}$
min_m	$=$	0 minden $m \in M$
max_m	$=$	$\begin{cases} \infty & m = s, d \\ 0 & \text{otherwise} \end{cases}$
cap_o	$=$	$w((v_1, v_2))$, where $o = (\{v_1\}, \{v_2\})$
fix_o	$=$	0 for all $o \in O$
$prop_o$	$=$	0 for all $o \in O$
$price_m$	$=$	$\begin{cases} 1 & m = d \\ 0 & \text{otherwise} \end{cases}$

Transportation problem

- Given are a set of supply sources S and destinations D, moreover:
  - For all $s \in S$ source, a produced amount of a product is given
  - For all $d \in D$ destination, a demand for the amount of the product is given
  - For each s,d pair, the proportional transportation cost is given
- The objective is to find a weight function -representing the transportation amounts - for the complete bipartite graph with partitions S and D such that
  - For all source $s \in S$ the aggregated weight of the adjacent edges does not exceed the produced amount.
  - For all $d \in D$ destination, the aggregated weight of the adjacent edges reach the demand
  - The wighted sum of the transportations costs is minimal
- The problem can efficiently be solved to optimality by the simplex algorithm

Solving the transportation problem with the P-graph framework

- Modelling
  - Materials correspond to sources and destinations
    - For all source $s \in S$ a raw material is assigned
    - For all destination $d \in D$ a product is assigned
    - The maximal limit for the raw material of each $s \in S$ is the production limit of s
    - The lower bound for the amount of the product for each $d \in D$ is the required amount of d
    - Prices and costs are not defined for the materials.
  - Operating units correspond to the transportation routes
    - For all $s \in S$ and $d \in D$ an operating unit ({s},{d}) is assigned
    - The proportional cost of the units is the proportional transportation cost between s and d
    - Capacity limits and fixed costs are not introduced.

Solving the transportation problem with the P-graph framework

M	$=$	$S \cup D$
R	$=$	S
I	$=$	$\emptyset$
P	$=$	D
O	$=$	$\{(\{s\},\{d\}) \mid s \in S, d \in D\}$
min_m	$=$	$\begin{cases} dem_m & m \in D \\ \infty & \text{otherwise} \end{cases}$
max_m	$=$	$\begin{cases} supp_m & m \in S \\ \infty & \text{otherwise} \end{cases}$
cap_o	$=$	∞ for all $o \in O$
fix_o	$=$	0 for all $o \in O$
$prop_o$	$=$	$cost_{s,d}$ where $o = (\{s\},\{d\})$
$price_m$	$=$	0 for all $m \in M$

Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- Cormen, T., Leiserson, C., Rivest, R., Stein, C., Introduction to Algorithms, The MIT Press, 2009.
- Bertok, B., Kovacs, Z., Gyártórendszerek modellezése

http:// tananyagfejlesztes.mik.uni-pannon.hu/

Modelling of production systems

Integration

Introduction

- Heat integration
- Integration of PNS and scheduling

Heat integration

Introduction

- The process contains heat stream
  - Cold streams need heating
  - Hot streams need cooling
- The cost consists of
  - Operating units cost
  - Raw materials cost
  - Heat exchanges cost

Heat exchanger network (HEN)

- The heating and cooling duties can be satisfied by
  - Hot or cold utilities
    - For example steam or water which have cost
  - Other heat stream
- Heat exchanger unit is necessary
  - Cost of heat exchange

Input of HEN

- The set of hot streams (F^{H})
- The set of cold streams (F^{C})
- The rate and the heat capacity of materials streams
- The set of heat sources (U^{H}) and heat sinks (U^{H}), their temperatures and costs
- The cost of heat exchangers

Heat stream of a material

- A material stream can have temperatures
- If the temperature of a material is different on different operating unit, it needs heat exchange
- The rate of enthalpy flow is proportional with the flow rate of the material

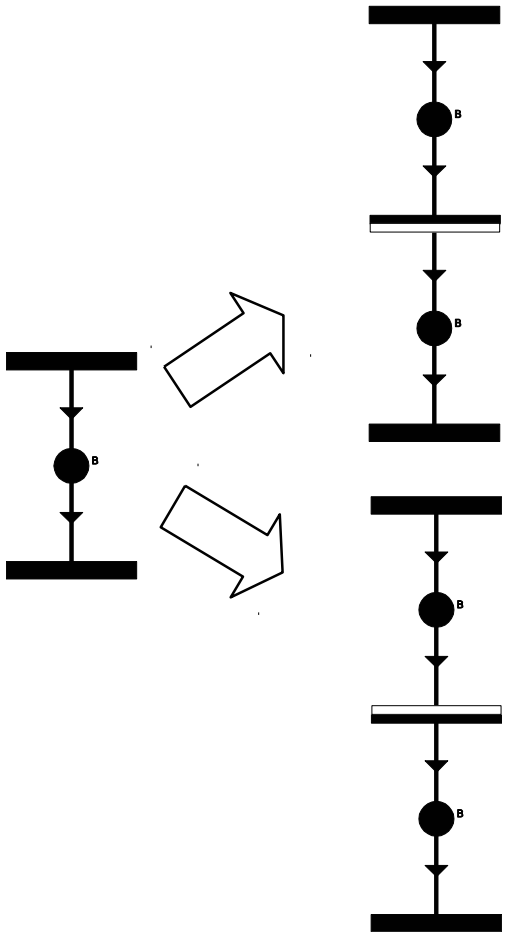
Latent heat

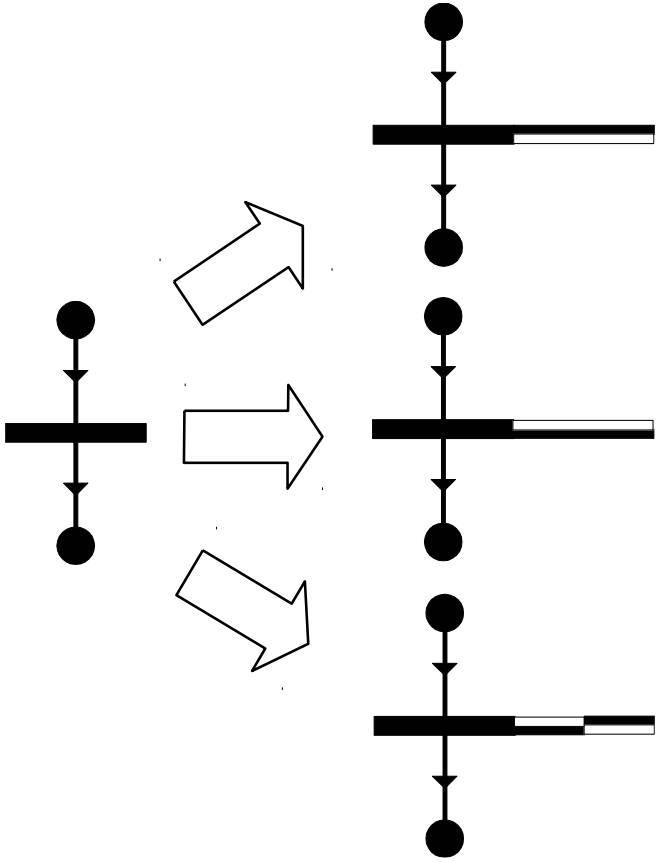
- An operating unit needs heating or cooling to remain its temperature constant
- The latent heat has temperature and rate of enthalpy flow
- An operating unit can have multiple latent heats
  - A part of the operating unit has cooling duty an other part has heating duty

hP-graph

- The hP-graph contains both operating and heat-exchanger units
- The node for a heat-exchanger unit for heating is indicating by a bar with solid lower half
  - For cooling, by a bar with a solid upper half
- When an operating unit has latent heat, the node for it is extended by an appropriate heat-exchanger unit

hP-graph

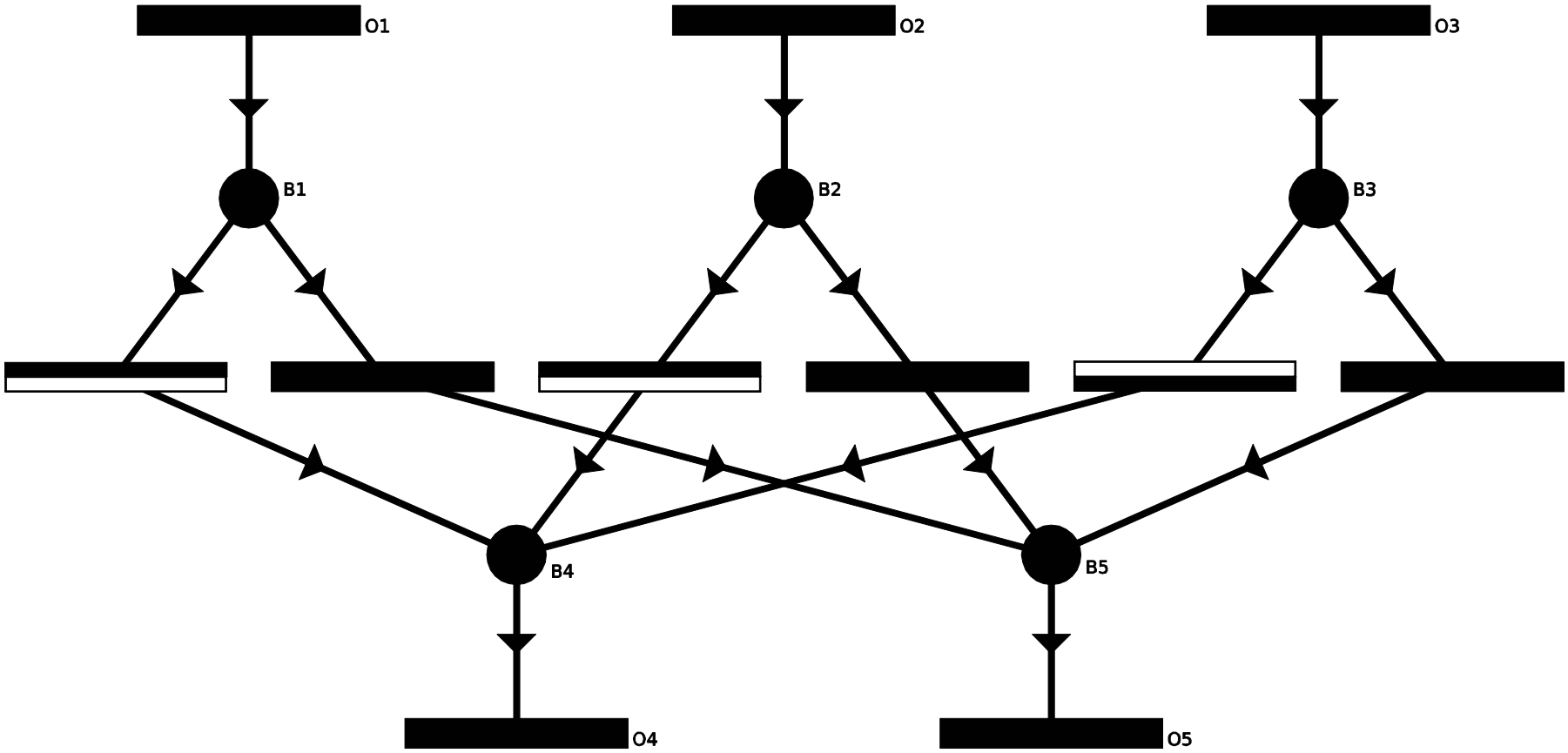




Extension of material nodes

- Heat streams with different temperatures cannot mixed
- In hP-graph each operating unit has it own material node
- Material flows are represented by fictive operating units or heat exchangers

Extension of a material node



Temperatures

- Let t_{ij}^{in} and t_{ij}^{out} denotes the input and the output temperature of material m_i for operating unit o_j, respectively
- Let l_j denotes the number of latent heats of operating unit o_j
- Let h_{ij} denotes the rate of latent heat i of operating unit o_j
  - $h_{ij} > 0$ if the operating unit releases heat
  - $h_{ij} < 0$ if the operating unit absorbs heat
- Let TM_{ij} denotes the temperature of the latent heat

Heat streams

- A material stream can be denoted by a triple (i, j, k)
  - The material (m_i)
  - The operating unit producing it (o_j)
  - The operating unit consuming it (o_k)
- A heat stream is such a material stream which has temperature parameters for both operating units

Heat streams

- The set of hot streams

$$\mathrm{F}^{H} = \{(i,j,k): t_{ij}^{out} > t_{ik}^{in}, m_i \in \mathcal{M}, o_j \in \varphi^{-}(m_i), o_k \in \varphi^{+}(m_i)\} = \{FH_1, FH_2, \ldots, FH_{n_{FH}}\}$$

- The set of cold streams

$$\mathrm{F}^{C} = \{(i,j,k): t_{ij}^{out} < t_{ik}^{in}, m_i \in \mathcal{M}, o_j \in \varphi^{-}(m_i), o_k \in \varphi^{+}(m_i)\} = \{FC_1, FC_2, \ldots, FC_{n_{FC}}\}$$

- For simplicity let $t_0^{(i,j,k)}$ and $t_1^{(i,j,k)}$ denotes t_{ij}^{out} and t_{ij}^{in} such that $t_0^{(i,j,k)} < t_1^{(i,j,k)}$

Latent heat

- A latent heat can be denoted by a pair (i, j)
  - The operating unit (o_j)
  - The number of the latent heat ($i = 0,1, \dots, l_j$)
- Set of cold heat sources

$$\mathrm{L}^H = \left\{(j,i): h_{ji} > 0, o_j \in \mathcal{O}, i \in \{1, \dots, l_j\}\right\} = \left\{LH_1, LH_2, \dots LH_{n_{LH}}\right\}$$

- Set of hot heat sources

$$\mathrm{L}^C = \left\{(j,i): h_{ji} < 0, o_j \in \mathcal{O}, i \in \{1,2, \dots, l_j\}\right\} = \left\{LC_1, LC_2, \dots LC_{n_{LC}}\right\}$$

- Let $t^{(i,j)}$ denotes the temperature of the latent heat (i, j)

Heat exchange

- Two heat streams, or a heat stream and a heat source, or two heat sources can exchange heat if their temperature intervals has common part
- A part is enough because a heat stream or a heat source can exchange heat several times with different heat streams and heat sources

Inherent temperature intervals

- Inherent temperature intervals are the narrowest intervals on which heat exchange can occur
- Sorting all temperatures in increasing order
  - If more heat stream or heat source have the same temperature it presents only once in the list
  - $t_1, t_2, \dots, t_{n_e+1}$, where if $i < j$ then $t_i < t_j$
- $E_p = [t_p, t_{p+1}],\ p \in \{1, \dots, n_e\}$ are the inherent temperature intervals

Inherent streams

- All heat streams are divided into inherent streams according to the inherent temperature intervals
- Hot inherent streams

$$\mathrm{L}^{H} = \{(i,j,k,p): (i,j,k) \in \mathrm{F}^{H}, t_p \geq t_0^{(i,j,k)}, t_{p+1} \leq t_1^{(i,j,k)}, p \in \{1,\dots,n_e\}\} = \{FSH_1, FSH_2, \dots FSH_{n_{FSH}}\}$$

- Cold inherent streams

$$\mathrm{L}^{C} = \{(i,j,k,p): (i,j,k) \in \mathrm{F}^{C}, t_p \geq t_0^{(i,j,k)}, t_{p+1} \leq t_1^{(i,j,k)}, p \in \{1,\dots,n_e\}\} = \{FSC_1, FSC_2, \dots FSC_{n_{FSH}}\}$$

Composite substreams

- Composite substreams are the merging of neighbor inherent streams of a heat stream
- If the (i, j, k) heat stream has $d-1$ inherent temperature intervals then $\binom{d+1}{2}$ subinterval exists
- Each subinterval has the form of $l_{qs} = [t_q, t_{s+1}]$, where $p \leq q \leq s \leq p + d - 1$

Composite substreams

- Hot composite substreams

$$\mathrm{I}^{H} = \{(i,j,k,q,s): (i,j,k) \in \mathrm{F}^{H}, I_{qs} \subseteq [t_0^{(i,j,k)}, t_1^{(i,j,k}]\} = \{SSH_1, SSH_2, \dots, SSH_{n_{SSH}}\}$$

- Cold composite substreams

$$\mathrm{I}^{C} = \{(i,j,k,q,s): (i,j,k) \in \mathrm{F}^{C}, I_{qs} \subseteq [t_0^{(i,j,k)}, t_1^{(i,j,k}]\} = \{SSC_1, SSC_2, \dots, SSC_{n_{SSC}}\}$$

Potential exchanges

- The potential exchanges of component substreams
- Define only for hot streams because it is symmetric

Potential exchanges

- Potential exchanges of hot component substreams with cold component substreams

$$JFF(SSH_l) = \{SSC_{l'} = (i', j', k', q', s') \in \mathrm{I}^C : q \leq q', s \leq s'\},\ SSH_l = (i, j, k, q, s) \in \mathrm{I}^H$$

- Potential exchanges of hot component substreams with cold latent heat

$$JFL(SSH_l) = \{LC_{l'} = (j', i') \in \mathrm{L}^C : t_q \leq T_{i'j'}\},\ SSH_l = (i, j, k, q, s) \in \mathrm{I}^H$$

Potential exchanges

- Potential exchanges of hot latent heats with cold component substreams

$$JLF(LH_l) = \{SSC_{l'} = (i', j', k', q', s') \in \mathrm{I}^C : T_{ij} \leq t_{s'+1}\},\ LH_l = (j, i) \in \mathrm{L}^H$$

- Potential exchanges of hot latent heats with cold latent heats

$$JLL(LH_l) = \{LC_{l'} = (j', i') \in \mathrm{L}^C : T_{ij} \leq T_{i'j'}\},\ LH_l = (j, i) \in \mathrm{L}^H$$

Heat exchanges with utility

- Potential exchanges of cold component substreams with hot utility

$$JFU(FSC_l) = \{u \in \mathrm{U}^H : UT_u \geq t_{p+1}\},\ FSC_l = (i,j,k,p) \in \mathrm{E}^C$$

- Potential exchanges of hot component substreams with cold utility

$$JFU(FSH_l) = \{u \in \mathrm{U}^C : UT_u \leq t_p\},\ FSH_l = (i,j,k,p) \in \mathrm{E}^H$$

Heat exchanges with utility

- Potential exchanges of cold latent heats with hot utility

$$JLU(LC_l) = \{u \in \mathrm{U}^H : UT_u \leq T_{ji}\},\ LC_l = (j,i) \in \mathrm{L}^C$$

- Potential exchanges of hot latent heats with cold utility

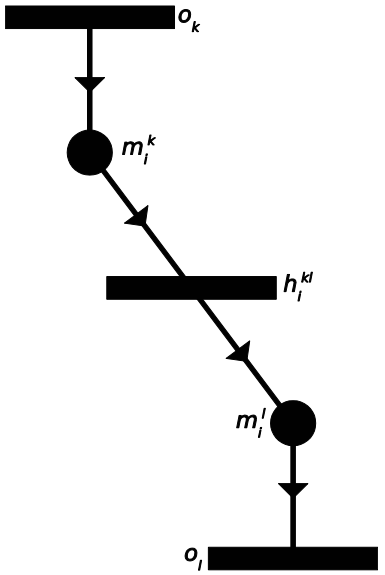
$$JLU(LH_l) = \{u \in \mathrm{U}^C : UT_u \geq T_{ji}\},\ LH_l = (j,i) \in \mathrm{L}^H$$

Mathematical model

- Extension of the linear PNS model
- The constraints does not change
  - Lower bounds on the amounts of products to be manufactured to meet the demand
  - Availability of raw materials
  - Mass balance

hP-graph

- T ($\subseteq \mathcal{M}$) denotes the sot of materials which have temperature
- Let $m_i \in$ T, $o_k \in \varphi^-(m_i)$ and $o_l \in \varphi^+(m_i)$
  - m_i^k and m_i^l denote the new material nodes belonging to o_k and o_l, respectively
  - t_{ik}^{out} and t_{il}^{in} denote the temperature of m_i^k and m_i^l, respectively
  - h_i^{kl} denotes the artificial operating unit of m_i from o_k to o_l
  - w_i^{kl} the amount of material go through h_i^{kl}



Constraints for new materials

- M_i and K_i denote the set of new materials and the set of new operating units for $m_i \in \mathrm{T}$, respectively
- Let $ir_{kj} = 1$ and $or_{kj} = 1$ for all $o_k \in \mathrm{U}_{i \in \mathrm{T}} \mathrm{K}_i$ and $j \in \psi(o_k)$
- Let φ'^{-} and φ'^{+} operator defines the operating units generating and consuming material of a hP-gráf, respectively

Constraints for materials

$$l_i \leq \sum_{o_k \in \varphi^-(i)} x_k \, or_{ki} - \sum_{o_k \in \varphi^+(i)} x_k ir_{ki} \leq u_i,$$
$$m_i \in \mathcal{M} \setminus \mathrm{T}$$

$$l_j \leq \sum_{o_k \in \varphi'^-(j)} x_k \, or_{kj} - \sum_{o_k \in \varphi'^+(j)} x_k ir_{kj} \leq u_j,$$
$$m_j \in \mathrm{M}_i, m_i \in \mathcal{M} \setminus \mathrm{T}$$

Heat transfer

- The rate of release or absorption of heat
  - Positive for hot streams and latent heats
  - Negative for cold streams and latent heats
- Heat stream

$$QFH_l = c_i w_i^{jk}\left(t_{p+1} - t_p\right), FSH_l = (i,j,k,p) \in \mathrm{E}^H$$

$$QFC_l = c_i w_i^{jk}\left(t_{p+1} - t_p\right), FSC_l = (i,j,k,p) \in \mathrm{E}^C$$

- Latent heat

$$QLH_l = h_{ji} x_j, LH_l = (j,i) \in \mathrm{L}^H$$

$$QLC_l = h_{ji} x_j, LC_l = (j,i) \in \mathrm{L}^C$$

Variables for heat transfer

- Define only for hot streams because it is symmetric
  - Nonnegative variables
- The first index denotes the hot and the second denotes the cold stream or latent heat
  - QFF_{ij}: $SSH_i \in \mathrm{I}^H$, $SSC_j \in JFF(SSH_i)$
  - QFL_{ij}: $SSH_i \in \mathrm{I}^H$, $LC_j \in JFL(SSH_i)$
  - QLF_{ij}: $LH_i \in \mathrm{L}^H$, $SSC_j \in JLF(LH_i)$
  - QLL_{ij}: $LH_i \in \mathrm{L}^H$, $LC_j \in JLL(LH_i)$

Variables for utility

- Nonnegative variables
- The order of indexes denotes the direction of heat transfer, i.e. the first index is the hot source, the second one is the cold source
  - QFU_{iu}: $FSH_i \in \mathrm{E}^H$, $u \in JFU(FSH_i)$
  - QUF_{ui}: $FSC_i \in \mathrm{E}^C$, $u \in JFU(FSC_i)$
  - QLU_{iu}: $LH_i \in \mathrm{L}^H$, $u \in JLU(LH_i)$
  - QUL_{ui}: $LC_i \in \mathrm{L}^C$, $u \in JLU(LC_i)$

Heat balance

- For each hot latent heat (LH_l), the rate of heat is the sum of heat transfer to cold streams, cold latent heats and cold utilities

$$QLH_l = \sum_{SSC_{l'} \in JLF(LH_l)} QLF_{ll'} + \sum_{LC_{l'} \in JLL(LH_l)} QLL_{ll'} + \sum_{u \in JLU(LH_l)} QLU_{lu},$$

$$LH_l \in \mathrm{L}^H$$

Heat balance

- For each cold latent heat (LC_l), the rate of heat is the sum of heat transfer from hot streams, hot latent heats and hot utilities

$$-QLC_l = \sum_{LC_l \in JFL(SSH_{l'})} QFL_{l'l} + \sum_{LC_l \in JLL(LH_{l'})} QLL_{l'l} + \sum_{u \in JLU(LC_l)} QUL_{ul},$$

$$LC_l \in \mathrm{L}^C$$

Heat balance

- For each hot stream ($FSH_l = (i, j, k, p)$) all composite substream must be taking into account

$$QFH_l = \sum_{SSH_n=(i,j,k,q,s)\in \mathrm{I}^H, s\le p\le q} \frac{t_{p+1} - t_p}{t_{q+1} - t_s} \left[\sum_{SSC_{n'}\in JFF(SSH_n)} QFF_{nn'} + \sum_{LC_{n'}\in JFL(SSH_n)} QFL_{nn'} \right] + \sum_{u\in JFU(FSH_l)} QFU_{lu}\,, FSH_l = (i, j, k, p) \in \mathrm{E}^H$$

Heat balance

- For each cold stream ($FSC_l = (i, j, k, p)$) all composite substream must be taking into account

$$-QFC_l = \sum_{SSC_n=(i,j,k,q,s)\in \mathrm{I}^C, s\le p\le q} \frac{t_{p+1} - t_p}{t_{q+1} - t_s}$$

$$\left[\sum_{SSC_n\in JFF(SSH_{n'})} QFF_{n'n} + \sum_{SSC_n\in JLF(LH_{n'})} QLF_{n'n}\right]$$

$$+ \sum_{u\in JFU(FSC_l)} QUF_{ul}\,, FSC_l = (i, j, k, p) \in \mathrm{E}^C$$

Heat streams heat transfer cost

- Suppose materials m_i and $m_{i'}$ exchange heat
  - $A_{ii'}$ denotes the unit cost of heat-transfer area
  - $U_{ii'}$ denotes the heat transfer coefficient
- The cost of heat exchange between composite substreams SSH_l and $SSC_{l'}$

$$CFF_{ll'} = A_{ii'} \frac{1}{U_{ii'} LMTD\left(t_s, t_{q+1}, t_{s'}, t_{q'+1}\right)}$$

$$SSH_l = (i, j, k, q, s) \in \mathrm{I}^H,$$

$$SSC_{l'} = (i', j', k', q', s') \in JFF(SSH_l)$$

LMTD

- Logarithmic mean temperature difference

$$LMTD(x_1, x_2, y_1, y_2) = \frac{(x_1 - y_1) - (x_2 - y_2)}{\ln \frac{x_1 - y_1}{x_2 - y_2}}$$

Latent heats heat transfer cost

- Let m a material which is used for heat transfer

$$CFL_{ll'} = A_{im}\frac{1}{U_{im}LMTD\left(t_s, t_{q+1}, T_{j'i'}, T_{j'i'}\right)},$$

$$SSH_l = (i, j, k, q, s) \in \mathrm{I}^H, LC_{l'} = (j', i')$$

$$CLF_{ll'} = A_{mi'}\frac{1}{U_{mi'}LMTD\left(T_{ji}, T_{ji}, t_{s'}, t_{q'+1}\right)},$$

$$LH_l = (j, i) \in \mathrm{L}^H, SSC_{l'} = (i', j', k', q', s') \in JFL(LH_l)$$

$$CLL_{ll'} = A_{mm}\frac{1}{U_{mm}}\frac{2}{\left(T_{ji} + T_{j'i'}\right)},$$

$$LH_l = (j, i) \in \mathrm{L}^H, LC_{l'} = (j', i') \in JLL(LH_l)$$

Utilities heat transfer cost

- In the model the cost of the heat transfer is linear
  - UC_u denotes the cost coefficient of utility u

Objective function

- The cost of the PNS and the cost of the heat exchange

$$\min \sum_{o_i \in \mathcal{O}} (fix_i y_i + prop_i x_i) + \sum_{r_j \in \mathcal{R}} \left(price_j \sum_{o_i \in \varphi^+(j)} x_i ir_{ij} \right)$$
$$+ \sum_{QFF_{jj'}} CFF_{jj'} QFF_{jj'} + \sum_{QFL_{jj'}} CFL_{jj'} QFL_{jj'}$$
$$+ \sum_{QLF_{jj'}} CLF_{jj'} QLF_{jj'} + \sum_{QFU_{iu}} UC_u QFU_{iu} + \sum_{QLU_{iu}} UC_u QLU_{iu}$$

Solution

- It can be solved by a modified ABB algorithm
- The branching do not change
- The bounding contains the new extended model
  - The operating units excluded from the structure are not presented in the model

PNS and scheduling

Introduction

- In PNS the operating units are continuous
- In the real life processes can be batch processes
  - They consume all input materials at the start and produce output only at the finish
- An operating unit can be used in different locations of the system without overlapping the operation in time
  - Scheduling

Scheduling

- The input of a scheduling problem can be defined by the structure of the process (recipe) and the set available equipment units
  - Multiple equipment units are available for a task
  - The operating time of a task depends on the assigned equipment unit
- An equipment unit must be assigned to each task
- An equipment unit cannot work on multiple tasks simultaneously
- Changeover time is the shortest time between two task of the same equipment
  - Cleaning, setup, …

Batch

- The production is based on batches
- To perform the recipe once generates one batch of a product
- If performing the recipe does not generate enough material, it has to be repeated
  - Multiple batches

Objective function

- Most common aims
  - Minimizing makespan for given amount of products
  - Maximizing profit in a given timehorizon
  - Minimizing earliness, tardiness
    - Due dates are given for products
  - Minimizing cost

S-graph framework

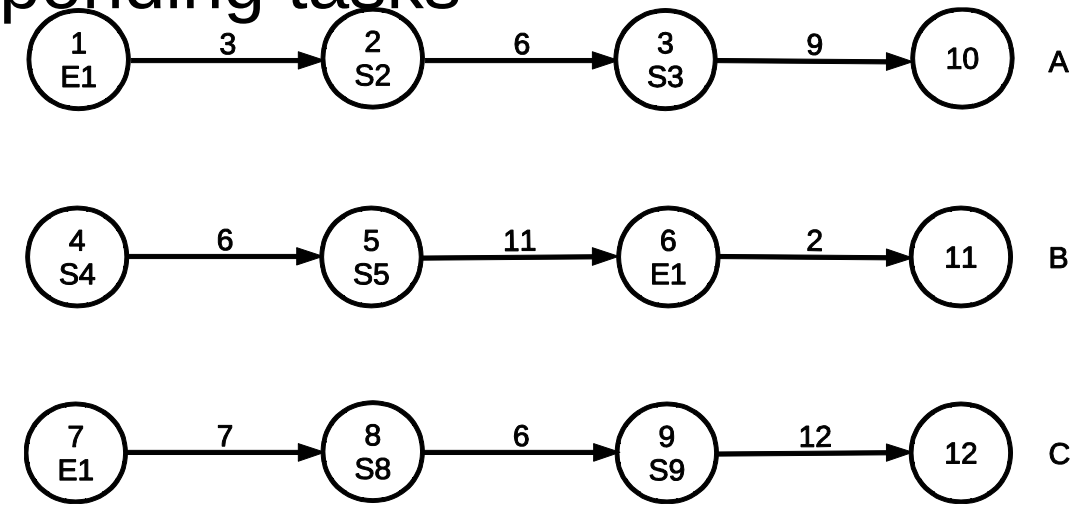
- To represent a scheduling problem and its solution we use a directed graph representation called S-graph
- A branch and bound algorithm can determine the optimal solution

Recipe-graph

- The recipe can be represented by a special S-graph, so called recipe-graph
- Nodes denote the tasks (task-nodes) and the products (product-nodes)
- Recipe-arcs denote the order of the tasks
  - The direction of the arcs are same as the direction of the material flow
  - The weight of a recipe-arc is the minimal difference of the starting time of the two connected task-nodes
    - If multiple equipment units are available the weight is equal to the shortest operation time

Example

- Three product (A, B, C) are to be produce
  - Three consecutive steps for each
- The sets S1, S2, …, S9 are the sets of available equipment units for the corresponding tasks

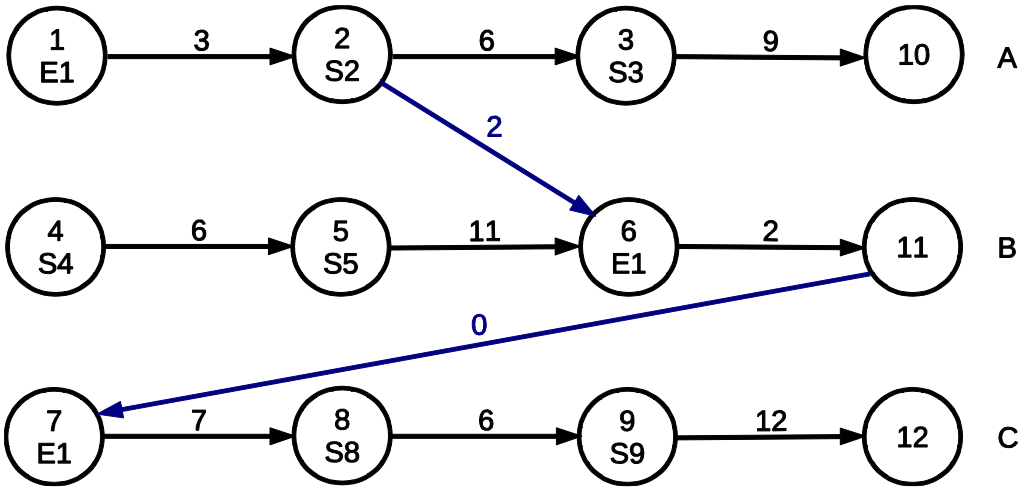


Scheduling on S-graph

- Scheduling
  - Assign an equipment unit for each task
  - Define a total order of the tasks of the same equipment unit
- Directed arcs (schedule-arcs) denote the operational order of equipment units
  - A schedule-arcs start from all the task-nodes following the actual node in the recipe-graph and point into the next task-node of the equipment unit
  - The weight of the arc is equal to the changeover time

Example

- Equipment unit E1 starts it work on task 1
- Fills its material into equipment unit assigned to task 2
- Continues its work on task 6 then task 7

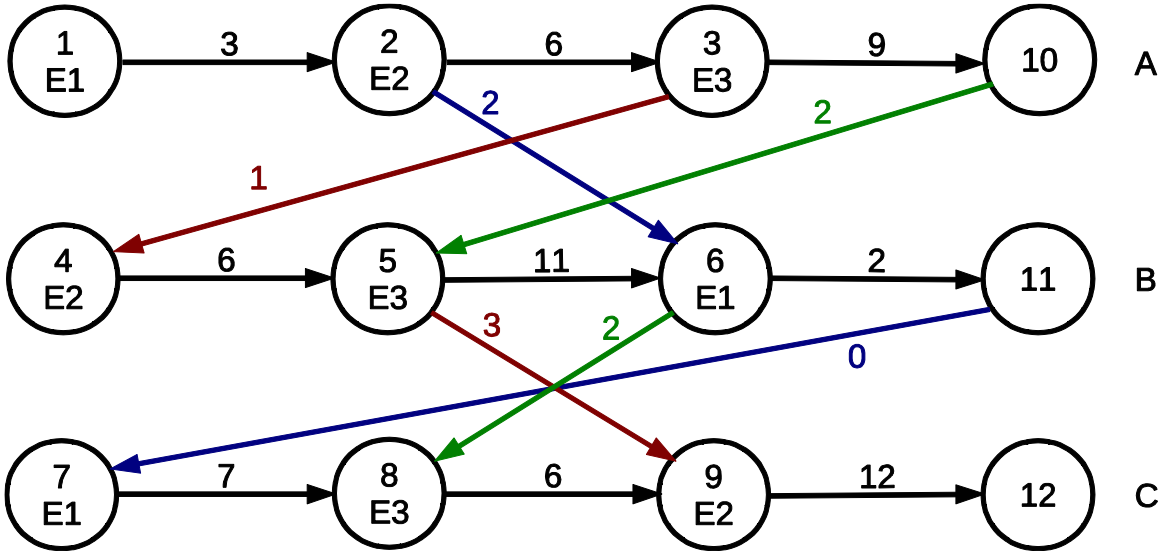


Schedule-graph

- An schedule-graph is a special acyclic S-graph which represents a solution
- There exists a unique schedule-graph for each solution
- In a schedule-graph all task (task-node) has been scheduled
  - According to the actual equipment-task assignment

Example

- Blue schedule-arcs belong to E1
- Red schedule-arcs belong to E2
- Green schedule-arcs belong to E3



B&B algorithm

- The most common aim is to minimize the makespan (the finishing time of the system)
  - The amount of product is given apriori
- The algorithm assign an equipment unit to a task in each step and determine its place in the activity list of the equipment unit
- The recipe-graph belongs to the root of the search tree
- A schedule-graph belongs to each leaf
- The bounding function is a longest path algorithm
  - For feasibility test it uses cycle search algorithm

Integrated problem

- PNS
  - Synthetize a process
- Scheduling
  - Schedule a given process
- Integrated problem
  - Synthetize a process which can generate all product in a given timehorizon

Integrated problem

- Operations (tasks) denotes the material transformations
  - Like operating units in PNS
- Operations can be performed by equipment units
  - Like in scheduling
- Aim
  - Determine the optimal structure
  - Give a feasible schedule in timehorizon

P-graph

- The P-graph of the synthesis problem is the base of the recipe
- The maximal structure must be acyclic
  - Cycles can be broken by introducing multiple batches

Integrated problem

- The equipment units assignment to operations affects the cost, the operating time and the capacity
- Cost are calculates from the costs of operations
  - Raw material costs are not taking into account
- Retrofit design
  - There are a set of available equipment units which have no investment cost
  - New equipment units can be purchased

Parameters

- The PNS parameters are the same
- Additional parameters are needed

Types of equipment units

- E is the set of equipment unit types
  - It contains the available and the purchasable types
- k_j is the number of available equipment units of type e_j
- $cost_j$ is the investment cost of an equipment unit of type e_j
- The costs, the operating times and the capacities are same for two equipment units of the same type

Plausible equipment units

- $OE(o_i)$ is the set of plausible equipment unit types for performing operation o_i
- Suppose that the cost of performing operation o_i with equipment unit type e_j is linear
  - a_{ij} denotes the proportional cost
    - The total cost is $a_{ij}x_i$ where x_i variable denotes the capacity of the operation o_i

Mass flow

- The mass flow on a equipment unit has bounds
- low_{ij} and upp_{ij} is the lower and the bound of operation o_i using equipment unit type e_j, respectively
  - $low_{ij} \leq x_i \leq upp_{ij}$

Operating time

- $time_{ij}$ is the operating time of operation o_i performing by an equipment unit of type e_j
  - It does not depend on time mass flow
- $ctime_{ii'j}$ is the changeover time of an equipment unit of type e_j between performing operation o_i and $o_{i'}$
- $TIME$ is the timehorizon

Solution procedure

- Based on ABB algorithm
- New decisions about equipment units
  - Assignment of equipment units to operations
    - Using an available equipment unit or buy new one
  - Scheduling

Search tree

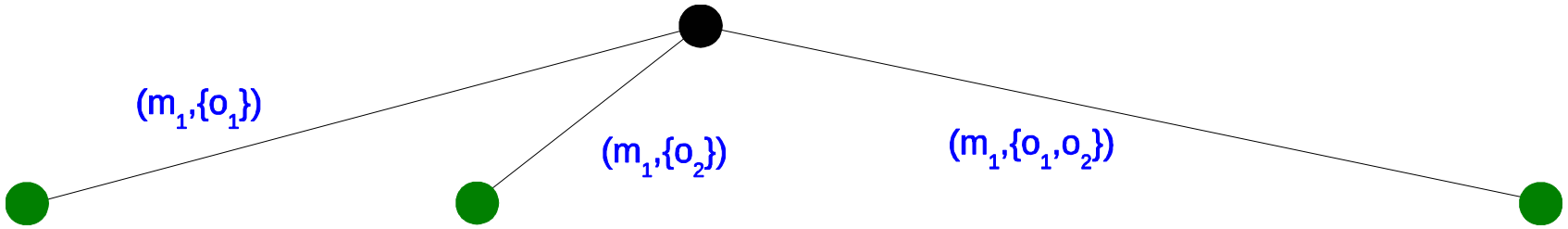
- First level
  - Original decisions of ABB
- Second level
  - Choosing of equipment unit type
- Third level
  - Decision about buying or not buying equipment unit

Example

- Let m_1 the current material
- It can be produced by operations o_1 and o_2
- Suppose that $OE(o_1) = \{e_1, e_3\}$, $OE(o_2) = \{e_2, e_3, e_4\}$
- There are available equipment units for each type

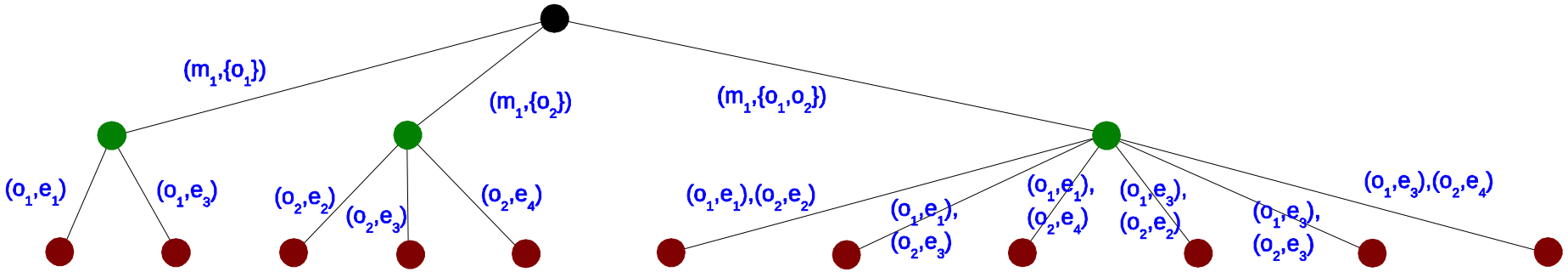
Example

- First level decisions
  - $(m_1, \{o_1\})$
  - $(m_1, \{o_2\})$
  - $(m_1, \{o_3\})$



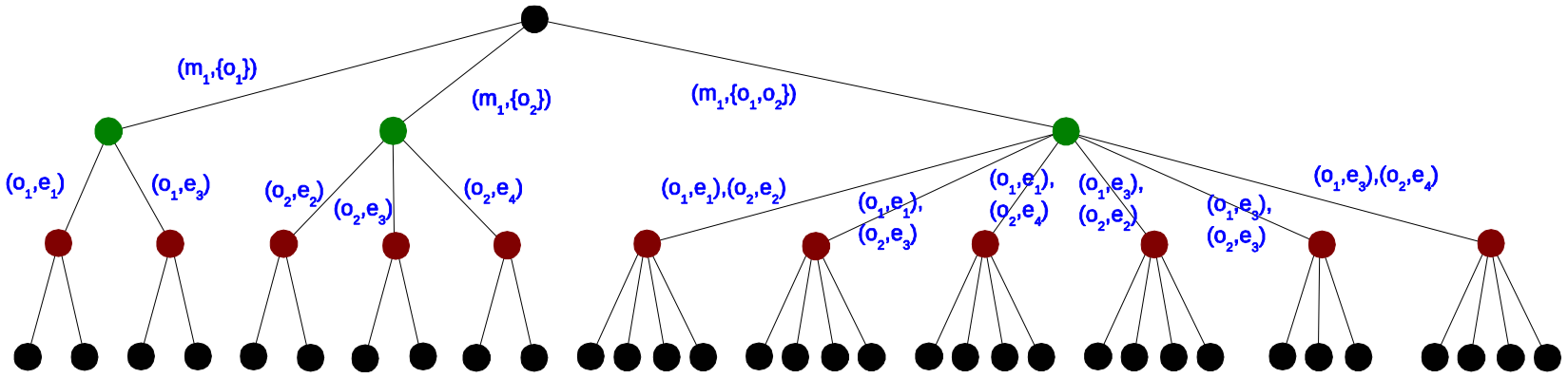
Example

- Multiple decision where multiple types are available for an operation
  - $|OE(o_1)| * |OE(o_2)| = 6$ decisions in the right side of the tree



Example

- Possible decisions
  - Two children – buying, not buying
  - Three children – buying one, buying two, not buying
  - Four children – buying two, buying for the first, buying for the second, not buying



Bounding

- Lower bound for cost
- Feasibility check with scheduling

Lower bound

- Relaxation of the PNS model extended by constraints
  - If operation o_i is excluded from the structure

$$0 \leq x_i \leq 0$$

  - If o_i is included in the structure and the type of equipment unit (e_j) is decided

$$low_{ij} \leq x_i \leq upp_{ij}$$

  - If o_i is included and type of equipment unit is not decided

$$\min_{e_j \in OE(o_i)} low_{ij} \leq x_i \leq \max_{e_j \in OE(o_i)} upp_{ij}$$

  - If there is no decision about o_i

$$0 \leq x_i \leq \max_{e_j \in OE(o_i)} upp_{ij}$$

Feasibility test

- Search for a scheduling of the structure with makespan less than timehorizon
- Scheduling works only in a fix structure
  - Solution-structure for the current decisions
- SSG algorithm can generate all solution-structures
- The generation stops when feasible schedule has been found
  - One feasible schedule is enough

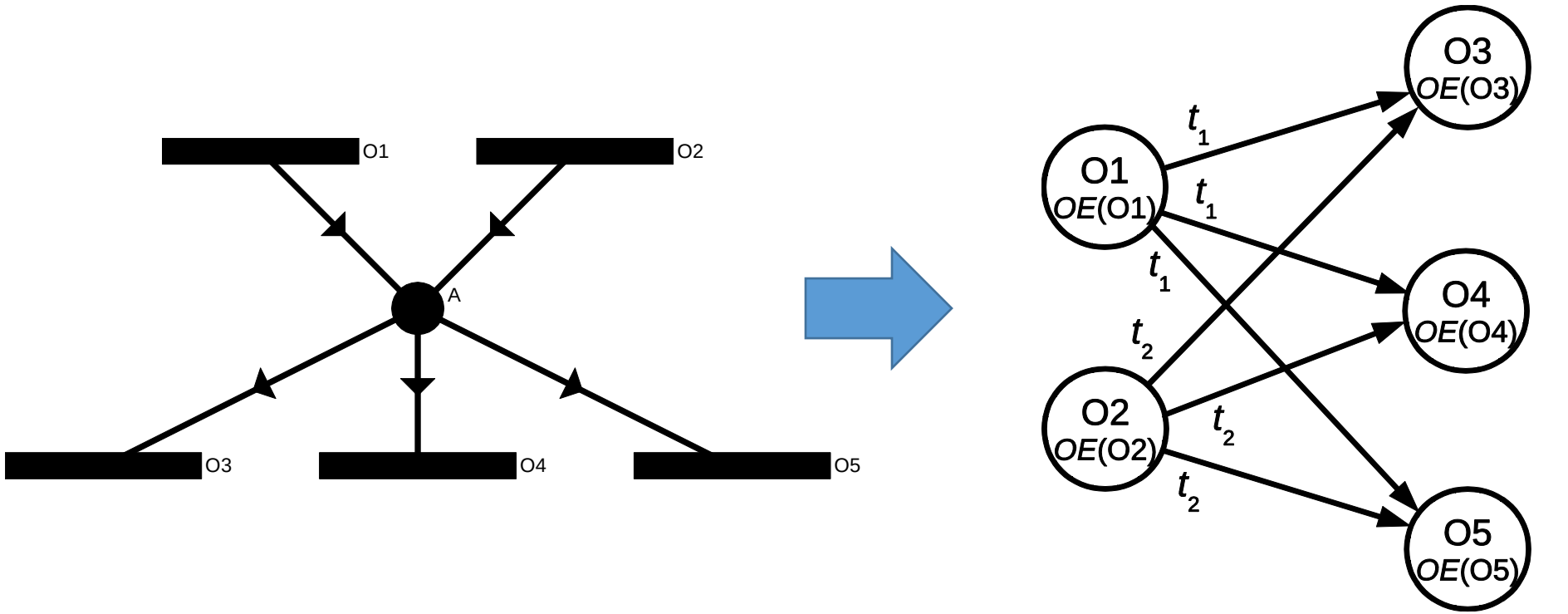
Scheduling of solution-structures

- Fictive tasks for undecided operations
  - No equipment unit for them
  - Do not need to schedule
  - Operating time is the smallest operating time of the operation
- Recipe-graph is necessary for scheduling
  - P-graph → recipe-graph

P-graph → recipe-graph

- One task-node for each operation
- One product-node for each product
- Recipe-arc for each connection
  - If o_i producing a material and o_j consuming it → recipe-arc from task-node of o_i to task-node of o_j
  - If o_i producing product → recipe arc from task-node of o_i to the corresponding product-node
  - The weight of the arc is the smallest operating time of the potential equipment units

Example


O1
O2
A
O3
O4
O5
O3
OE(O3)
t1
O1
OE(O1)
t1
t1
O4
OE(O4)
t2
O2
OE(O2)
t2
t2
O5
OE(O5)

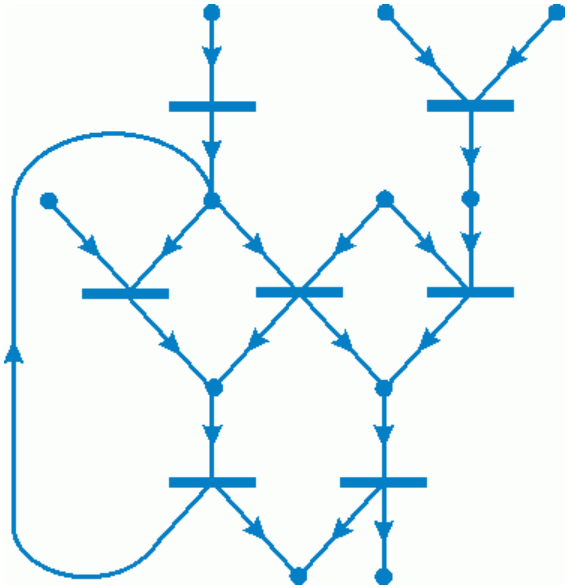
Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- M. A. Duran and I. E. Grossmann, Simultaneous optimization and heat integration of chemical processes, AIChE J. 32 (1986), 123–138.
- C. A. Floudas, A. R. Ciric, and I. E. Grossmann, Automatic synthesis of optimum heat exchanger network configurations., AIChE J. 32 (1986), 276–290.
- Sanmartí, E., T. Holczinger, L. Puigjaner, F. Friedler, Combinatorial Framework for Effective Scheduling of Multipurpose Batch Plants, AIChE Journal, 48(11), 2557-2570, 2002.
- Adonyi, R., J. Romero, L. Puigjaner, and F. Friedler, Heat Integration for Batch Processes presented at the PRES 2003, Hamilton, Ontario, Canada, October 26-29, 2003.

PNS Software Tools: PNS Draw and PNS Studio

Table of Contents

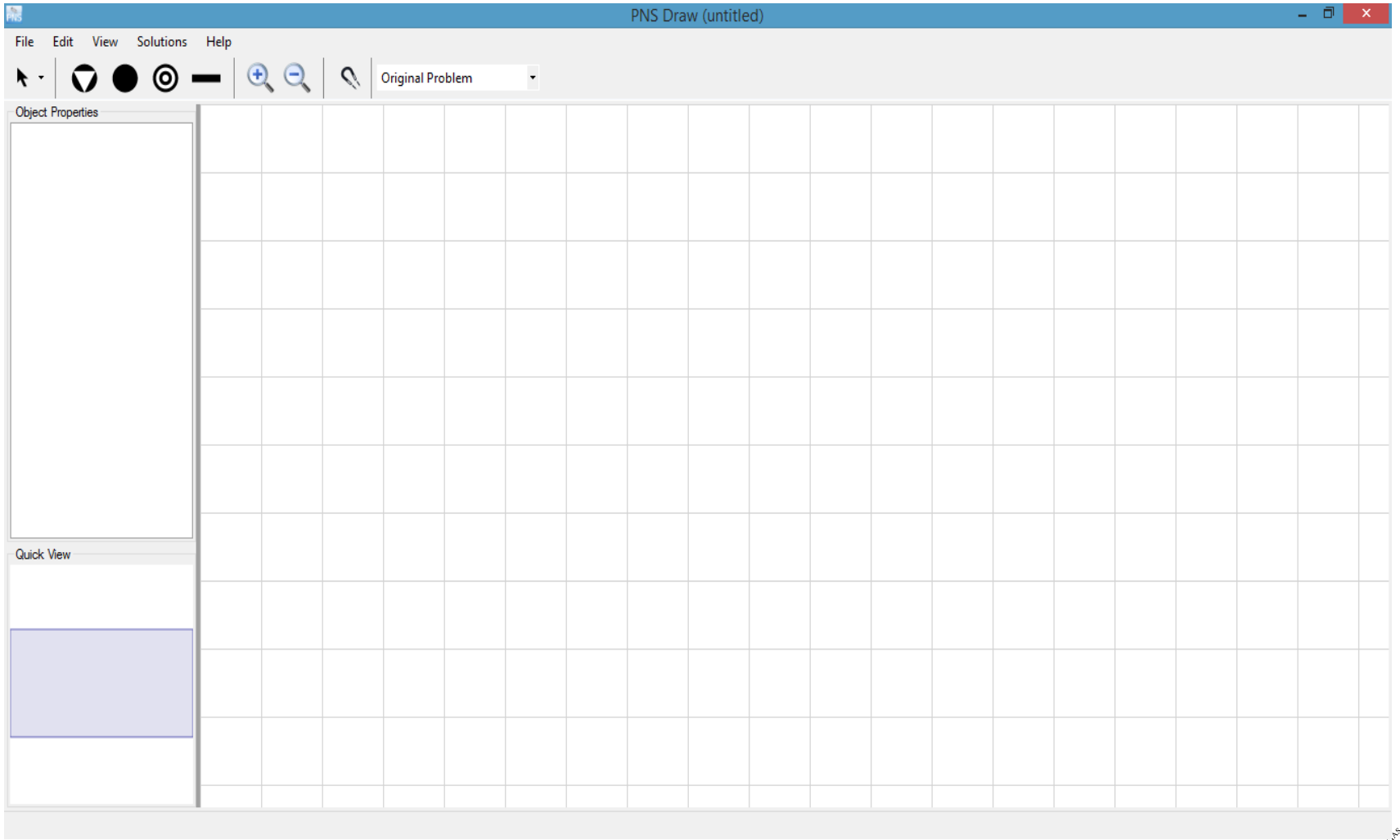
- PNS Draw
- PNS Studio
- Example



Introduction

- P-graph methodology is a good approach for
  - Process Design
  - Process Optimization
  - Flowsheet Optimization
  - Process Systems Engineering
  - Process Network Synthesis (PNS)
- The P-graph algorithms are supported by software tools
  - PNS Draw
  - PNS Studio

PNS Draw


PNS Draw (untitled)
File
Edit
View
Solutions
Help
Original Problem
Object Properties
Quick View

PNS Draw

- System Requirements
  - P3 800 CPU
  - 256 MB RAM
- Supported Windows Platforms:
  - Windows 2000 with Service Pack 3 and .NET 2.0
  - Windows XP with Service Pack 2 and .NET 2.0
  - Windows Vista
  - Windows 7
  - Windows 8

Features

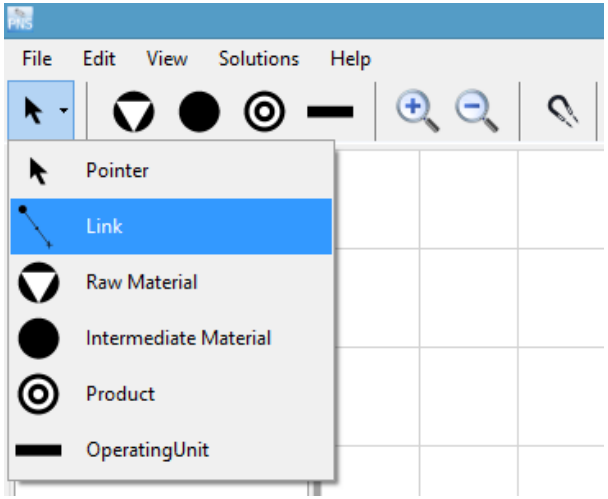
- Draw materials, operating units, connections
- Edit connection arrow position and define breakpoints and Bezier curves
- For PNS : define object's name and flow rates
- Define colors for objects
- Multiple object moving
- Zoom
- Grid and align to grid
- Export P-graph to PNG image format, SVG vector graphics format, PNS Studio format
- One step Undo / Redo

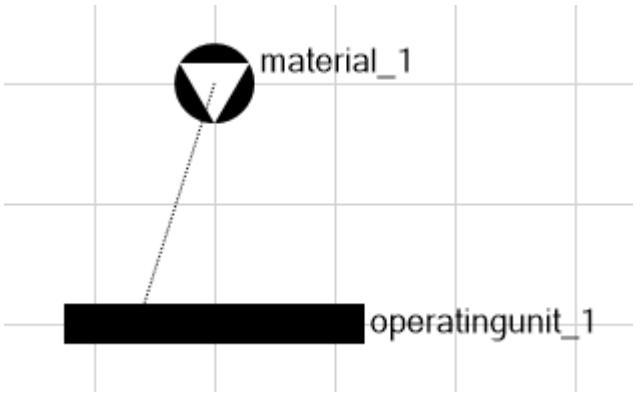
Using PNS Draw

- Adding materials and operating units
  - First option: Drag the material or operating unit symbol in the toolbar and drop to the editor area.
  - Second option: Change the draw mode in the toolbar to material or operating unit and click on the editor area to put them.

Using PNS Draw

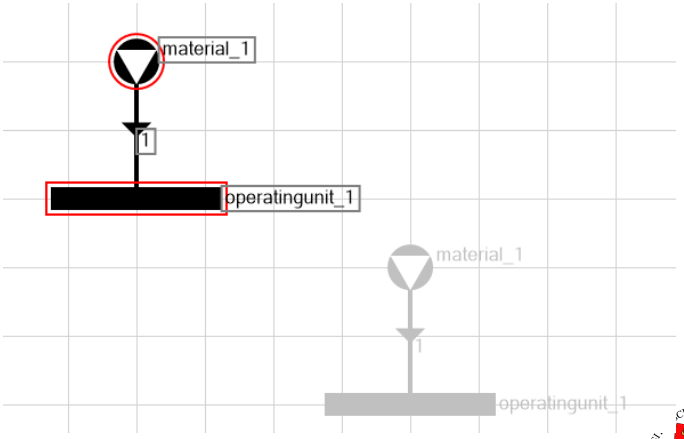
- Connecting objects
  - Select the Link option in the Drawing Mode dropdown list.
  - Click on the first object then click on the second object.





Using PNS Draw

- Selecting objects
  - Click on the object to select one object.
  - Press SHIFT key and click on another object to add to the selection, or remove from selection.
  - Press down the left mouse button and move the mouse to select the objects. You can use the SHIFT key to add or remove objects from selection.

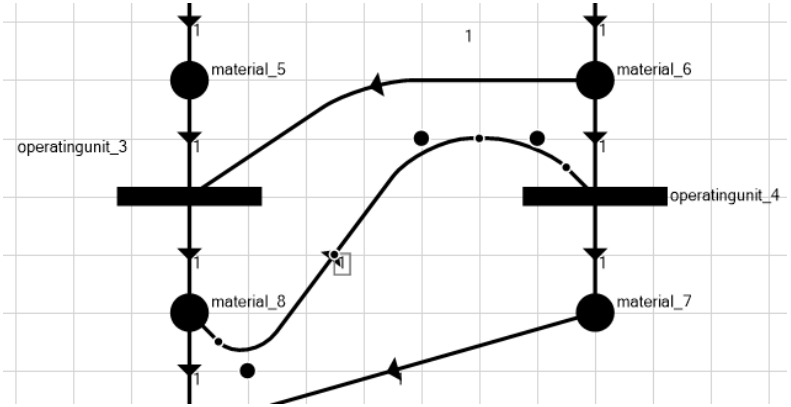


Using PNS Draw

- Scroll and zoom
  - Zoom: Ctrl + mouse wheel
  - Scroll Up-Down: mouse wheel
  - Scroll Left-Right: SHIFT + mouse wheel
  - Move the editor area: press down the right mouse button and move the mouse

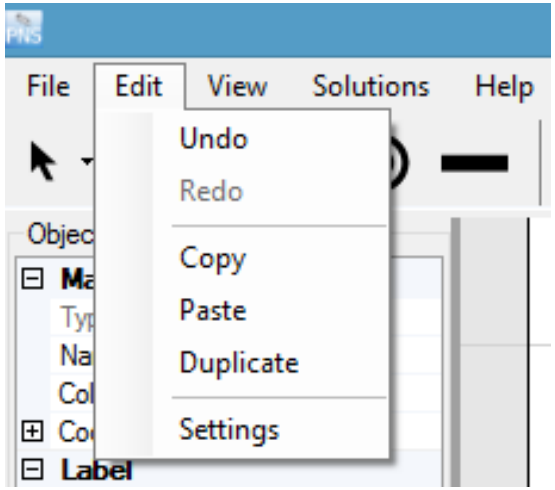
Using PNS Draw

- Add and remove breakpoints to lines
  - There are two breakpoints:
    - temporary (small dot)
    - line breakpoint (bigger dot)
  - Adding: Select a line (with left mouse button), the temporary breakpoints will be shown then move these to create new line breakpoints.
  - Removing: Select a line then select the line breakpoint then press the Delete key.

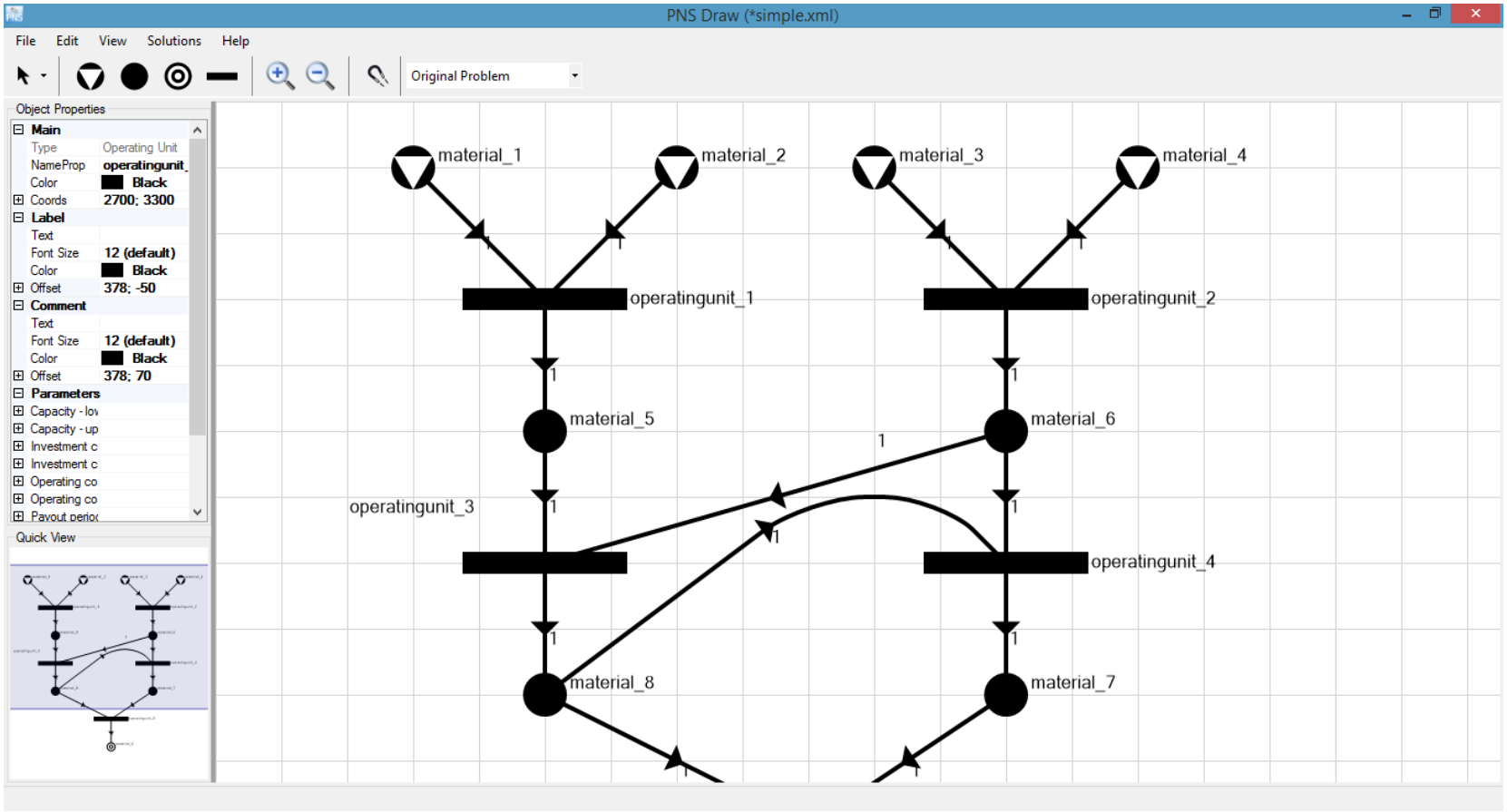


Using PNS Draw

- Keys
  - Ctrl + U: undo
  - Ctrl + Y: redo
  - Ctrl + S: save
  - Ctrl + C: copy
  - Ctrl + V: paste
  - Ctrl + D: duplicate
  - Delete: delete selected objects
  - Shift: invert selection
  - Alt: free moving of objects (when snap to grid is on)



PNS Draw Screenshot



Download link: http://www.p-graph.com/pnsdraw/

PNS Draw XML Output File Format

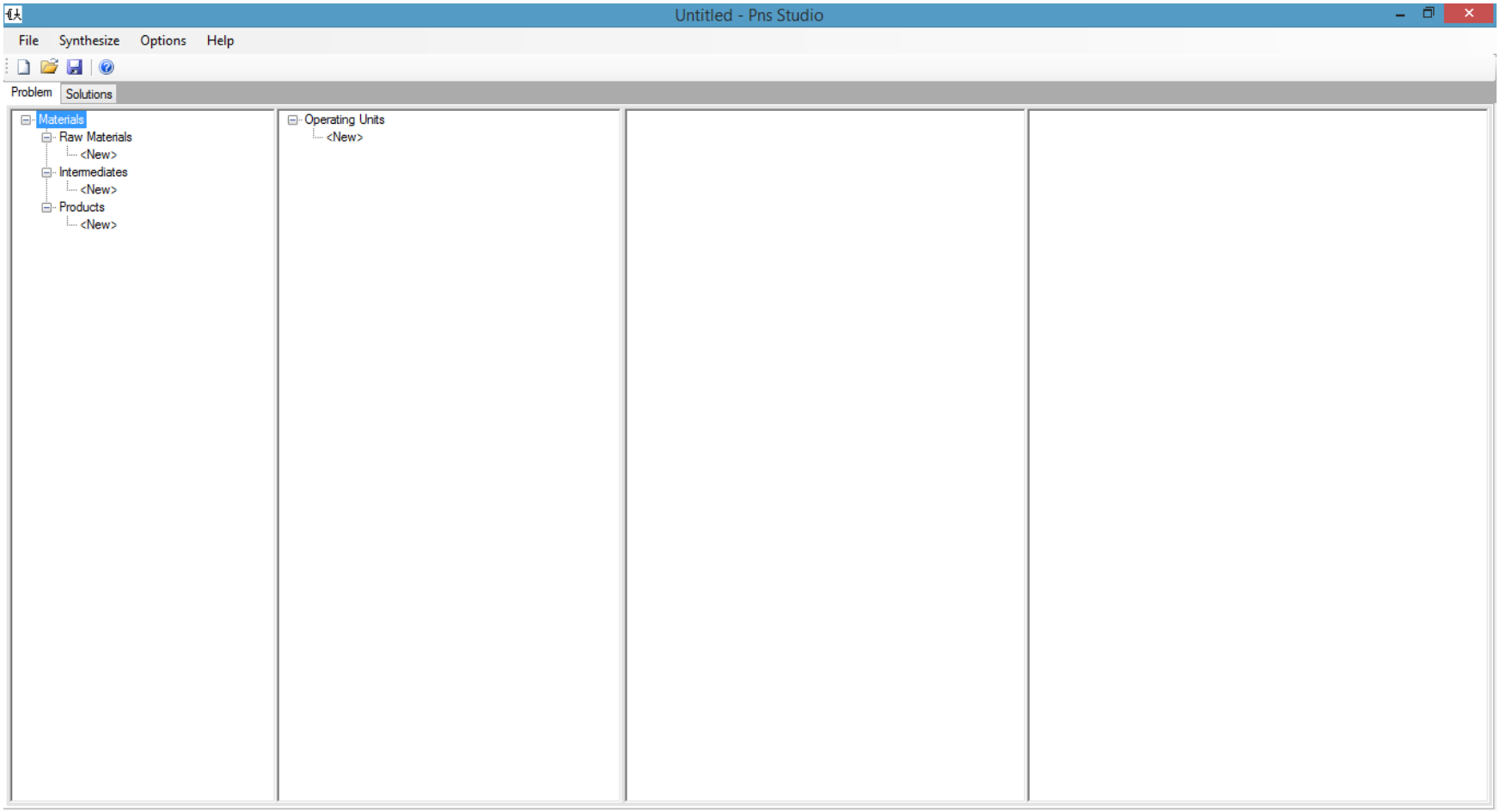
```
<Material ID="1" Name="material_1" Title="" Type="0">
      <ParameterList>
        <Parameter Name="price" Prefix="Price: " Value="-1" MU="" Visible="false" />
        <Parameter Name="reqflow" Prefix="Required flow: " Value="-1" MU=""
                                                                Visible="false" />
        <Parameter Name="maxflow" Prefix="Maximum flow: " Value="-1" MU=""
                                                                Visible="false" />
      </ParameterList>
      <Coords>
        <X>900</X>
        <Y>300</Y>
      </Coords>
      <Label Text="">
        <Offset>
          <X>103</X>
          <Y>-100</Y>
        </Offset>
        <FontSize>-1</FontSize>
        <Color>-16777216</Color>
      </Label>
          ...
```

PNS Draw XML Output File Format

```
<OperatingUnit ID="3" Name="operatingunit_1" Title="">
      <ParameterList>
        <Parameter Name="caplower" Prefix="Capacity, lower bound: " Value="-1" MU=""
                                                                    Visible="false" />
        <Parameter Name="capupper" Prefix="Capacity, upper bound: " Value="-1" MU=""
                                                                    Visible="false" />
        <Parameter Name="investcostfix" Prefix="Investment cost, fix: " Value="0" MU=""
                                                                    Visible="false" />
        <Parameter Name="investcostprop" Prefix="Investment cost, proportional: " Value="0" MU=""
                                                                    Visible="false" />
        <Parameter Name="opercostfix" Prefix="Operating cost, fix: " Value="0" MU=""
                                                                    Visible="false" />
        <Parameter Name="opercostprop" Prefix="Operating cost, proportional: " Value="0" MU=""
                                                                    Visible="false" />
        <Parameter Name="payoutperiod" Prefix="Payout period: " Value="-1" MU="" Visible="false„
                                                                                 />
        <Parameter Name="workinghour" Prefix="Working hours per year: " Value="-1" MU=""
                                                                    Visible="false" />

   </ParameterList>
      <Coords>
        <X>900</X>
        <Y>900</Y>
      </Coords>
          ...
```

PNS Studio

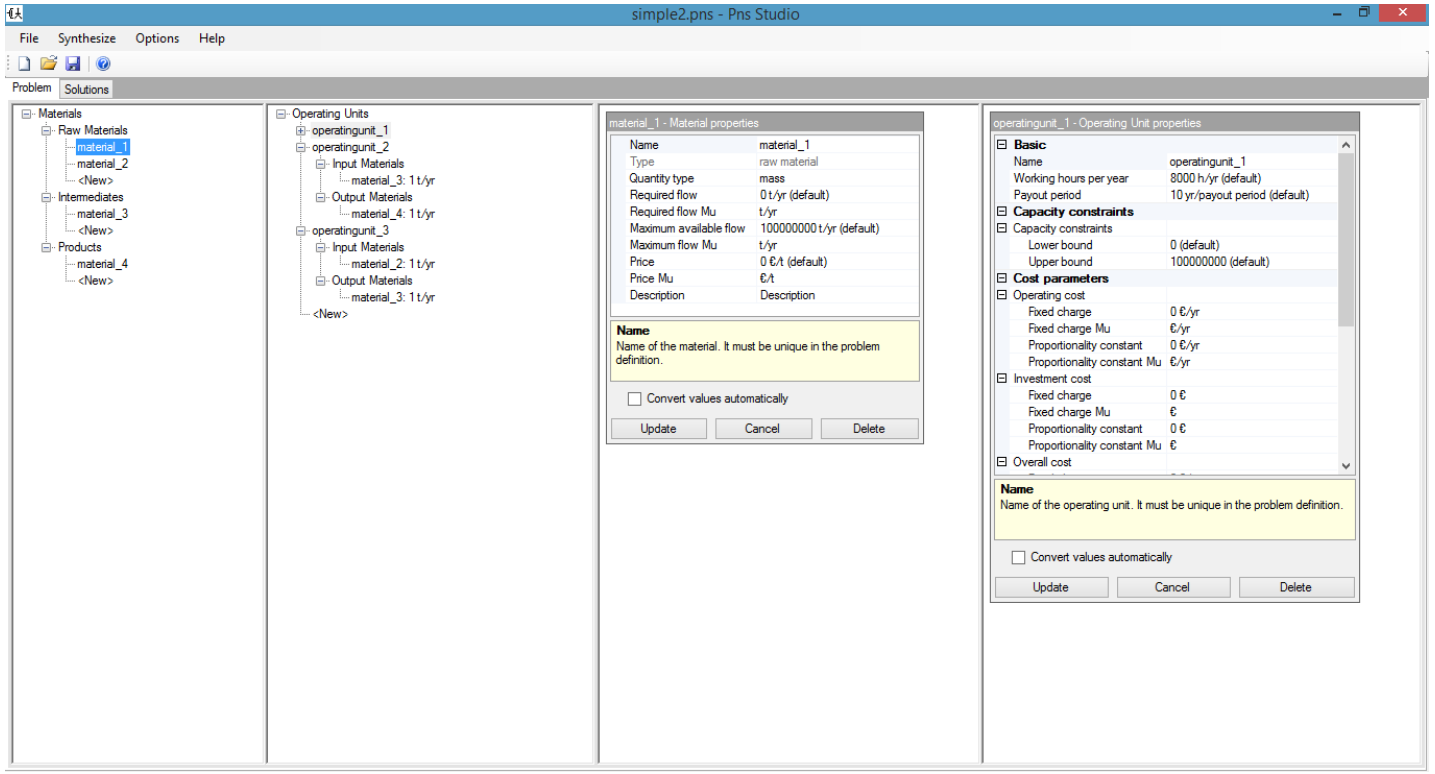

Untitled - Pns Studio
File
Synthesize
Options
Help
Problem
Solutions
Materials
Raw Materials
<New>
Intermediates
<New>
Products
<New>
Operating Units
<New>

PNS Studio

- System Requirements
  - At least P3 800 CPU
  - At least 256 MB RAM
- Supported Windows Platforms:
  - Windows 2000 with Service Pack 3 and .NET 2.0
  - Windows XP with Service Pack 2 and .NET 2.0
  - Windows Vista
  - Windows 7
  - Windows 8

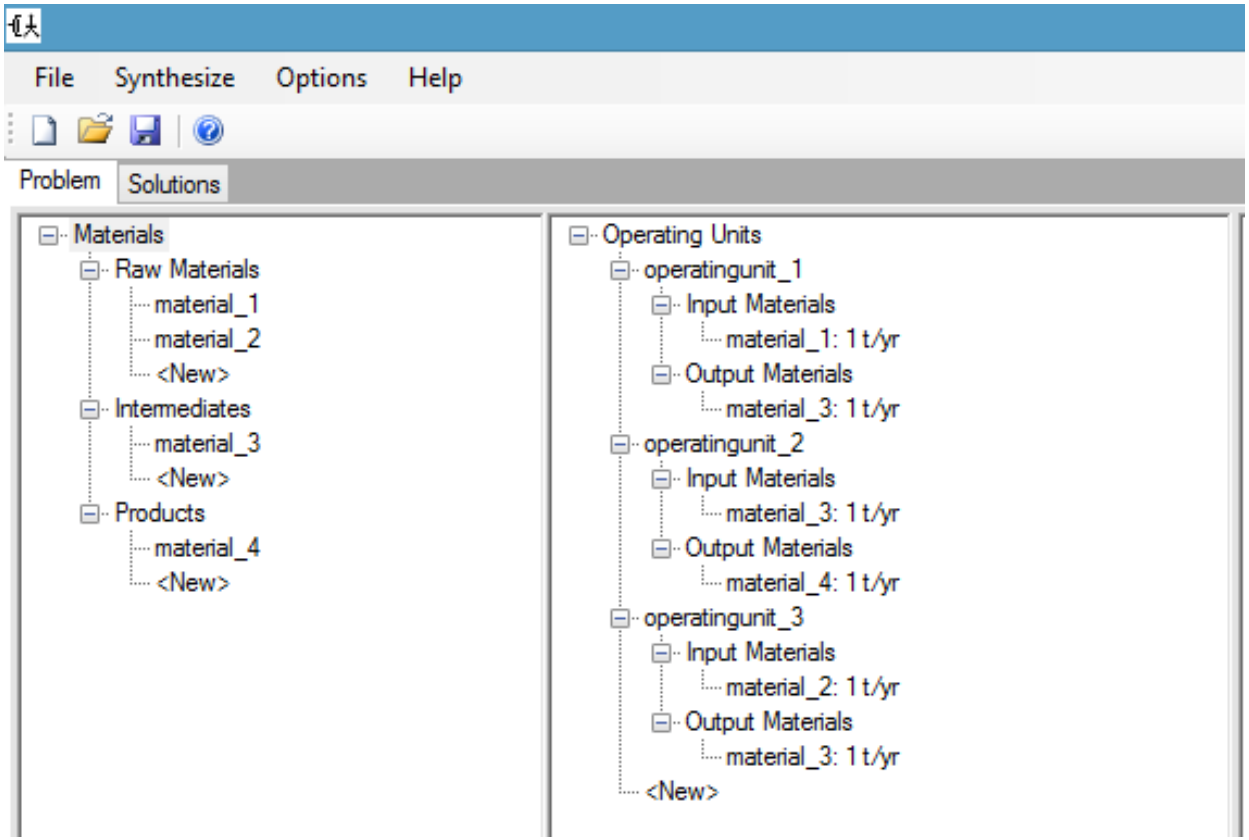
PNS Studio

- 4 parts of window: materials, operating units, parameters of materials, paramters of operating units



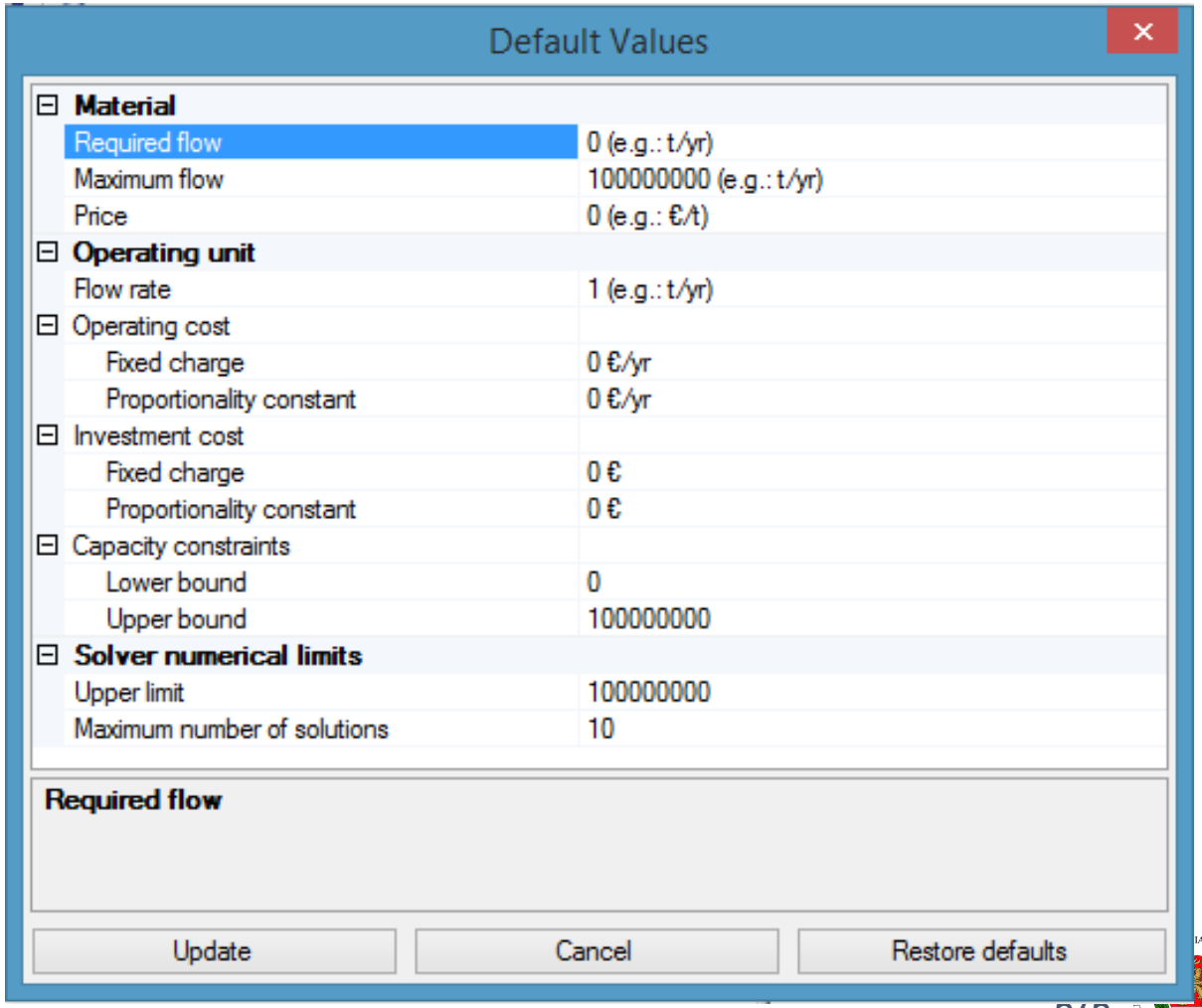
PNS Studio: Materials

- Materials: raw materials, intermediates, and products
- Operating units with input and output materials



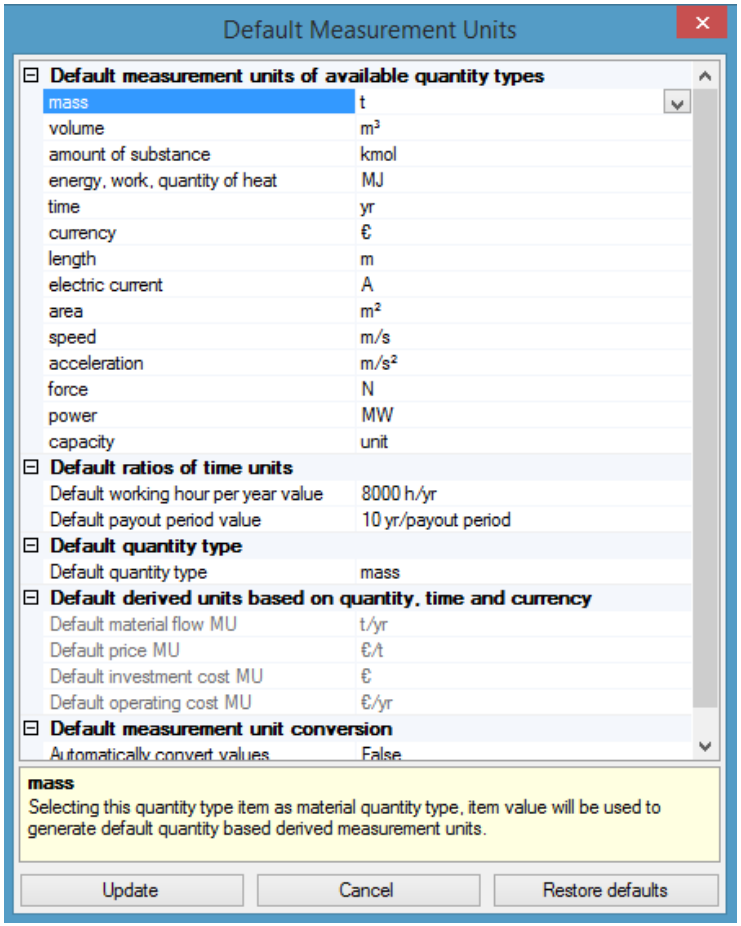
PNS Studio: Default values

- Options menu
  → Default Values

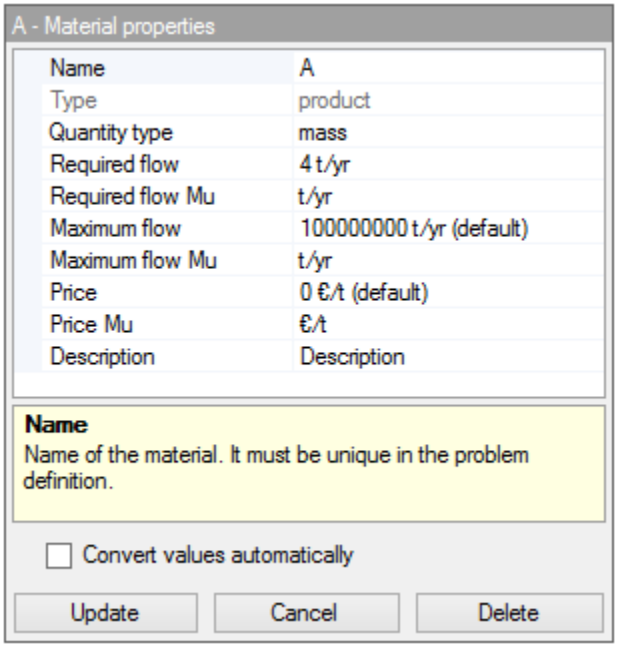


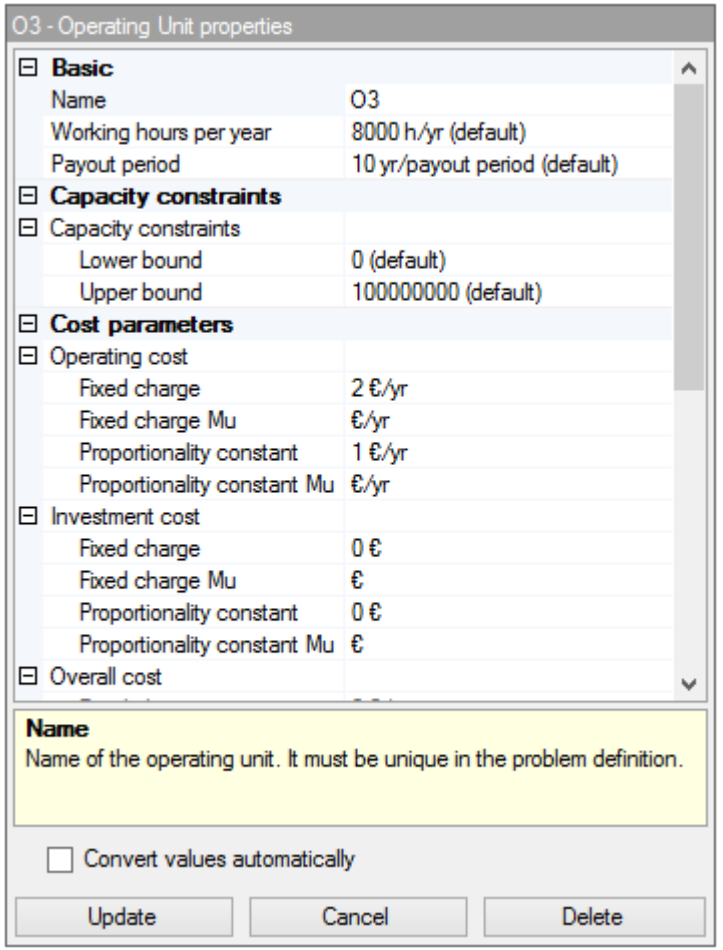
PNS Studio: Default Measurement Units

- Options menu
  → Default Measurement Units

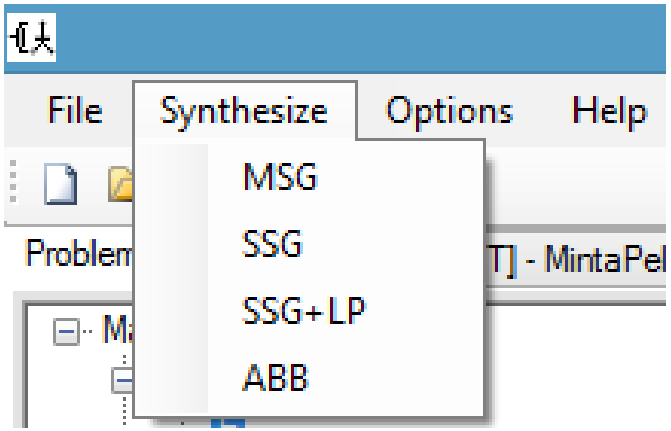


PNS Studio: Parameters of materials and operating units



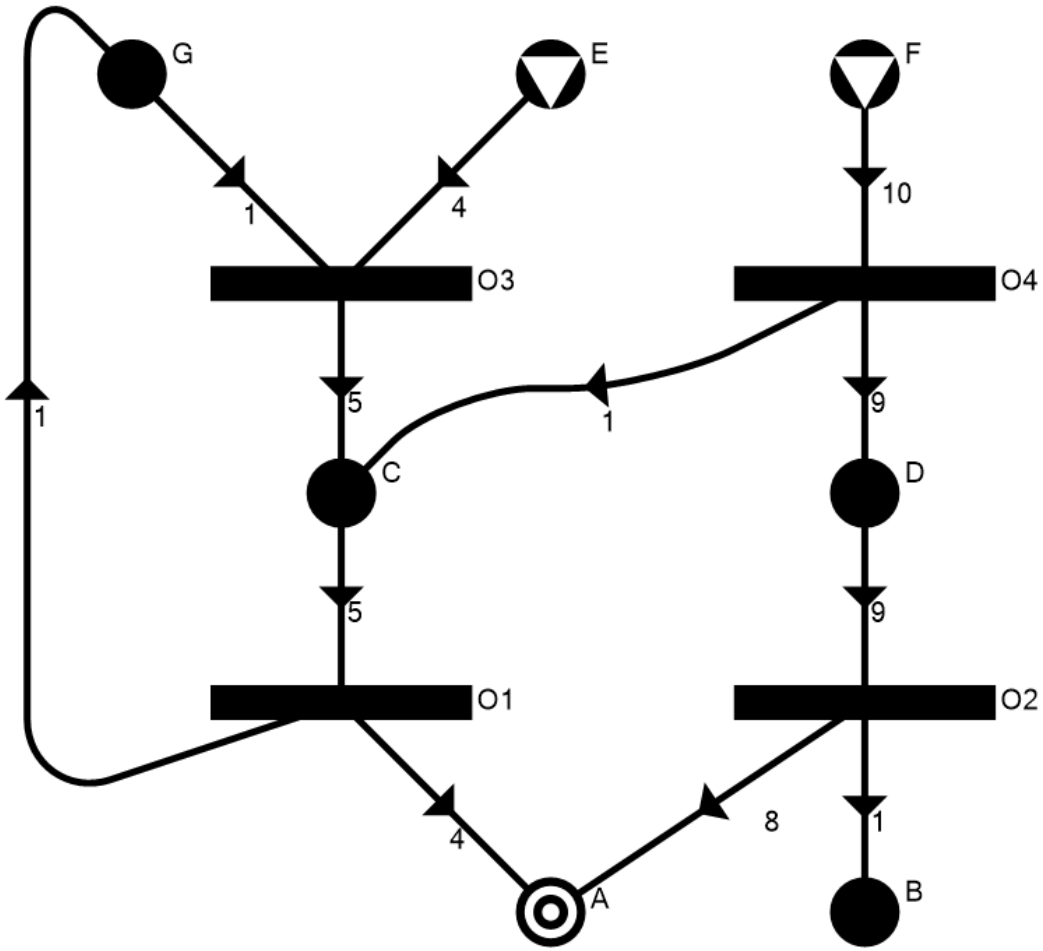


PNS Studio: P-graph algorithms

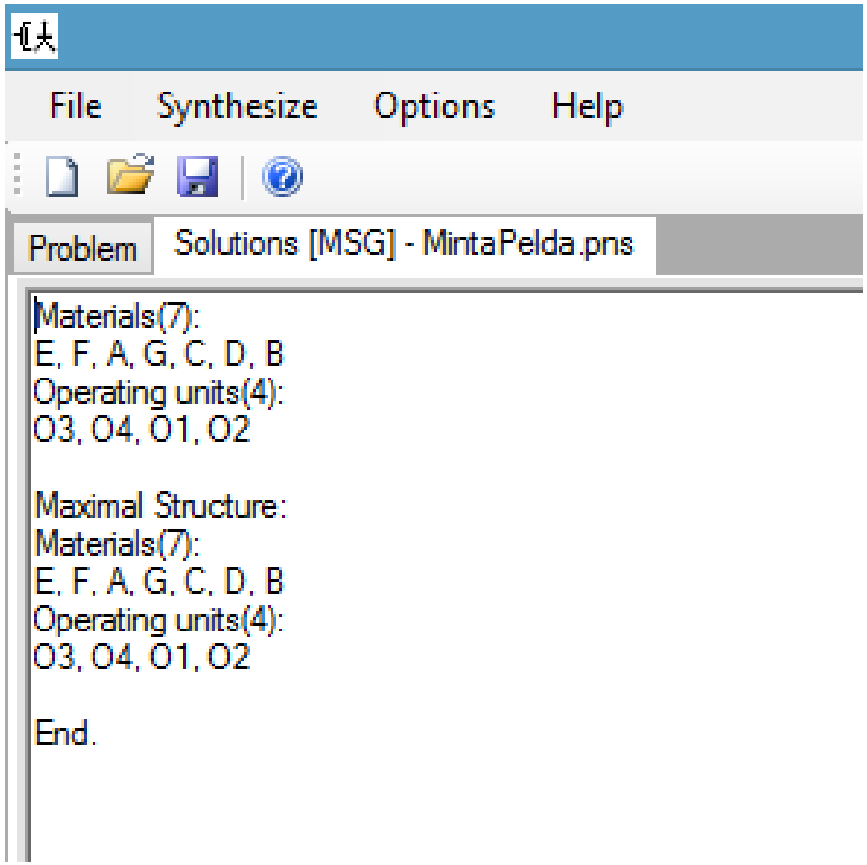


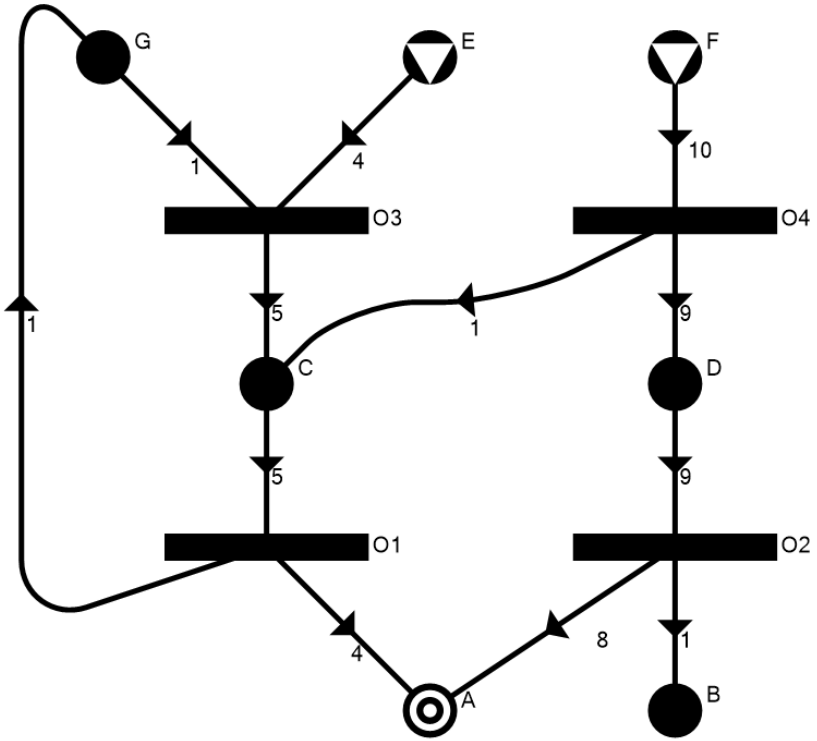
- Options menu
  → Synthesize menu
- MSG is the abbreviation of Maximal Structure Generation, which is an algorithm for PNS problems that determines maximal structure for the problem in polynomial time
- SSG is an algorithm that determines the all feasible solution structure for a PNS problem
- SSG+LP and ABB: To get the optimal structure we have to run the SSG+LP or ABB algorithm

PNS Studio: Example

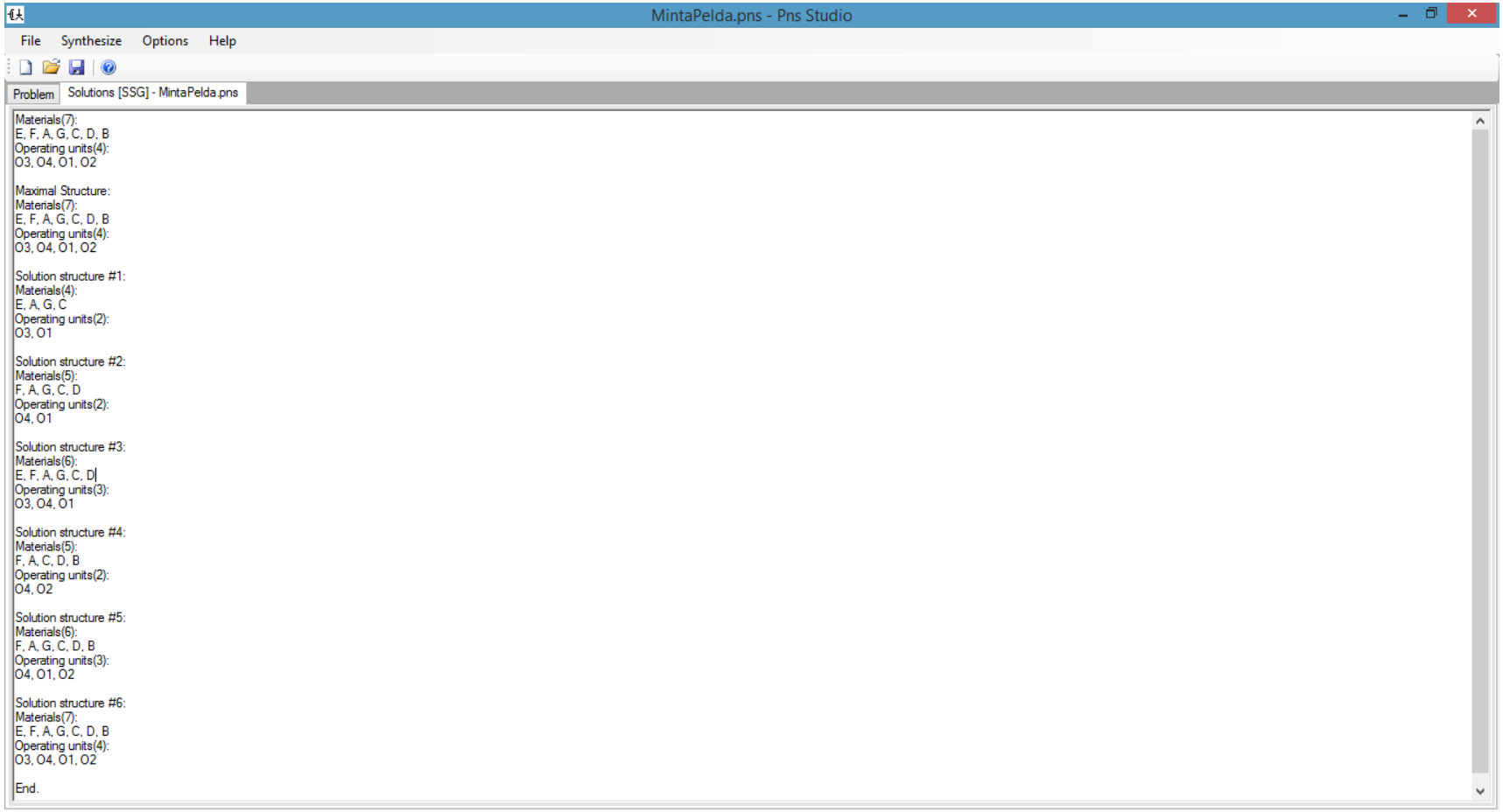


PNS Studio: Result of MSG algorithm





PNS Studio: Result of SSG algorithm



PNS Studio: Result of SSG algorithm

```
Materials(7):
E, F, A, G, C, D, B
Operating units(4):
O3, O4, O1, O2

Maximal Structure:
Materials(7):
E, F, A, G, C, D, B
Operating units(4):
O3, O4, O1, O2

Solution structure #1:
Materials(4):
E, A, G, C
Operating units(2):
O3, O1

Solution structure #2:
Materials(5):
F, A, G, C, D
Operating units(2):
O4, O1

Solution structure #3:
Materials(6):
E, F, A, G, C, D
Operating units(3):
O3, O4, O1

Solution structure #4:
Materials(5):
F, A, C, D, B
Operating units(2):
O4, O2

Solution structure #5:
Materials(6):
F, A, G, C, D, B
Operating units(3):
O4, O1, O2

Solution structure #6:
Materials(7):
E, F, A, G, C, D, B
Operating units(4):
O3, O4, O1, O2

End.
```

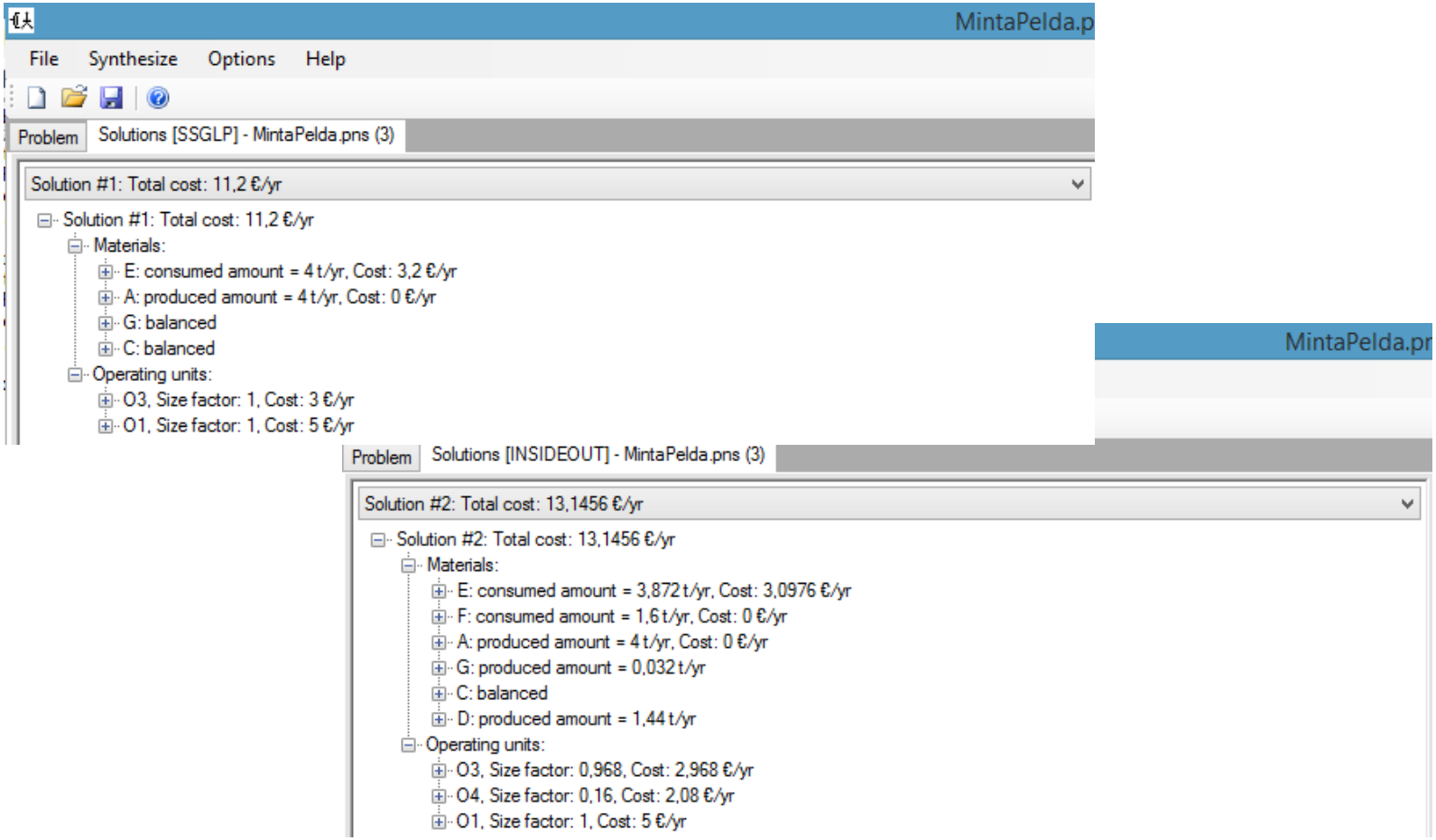
PNS Studio: Parameters for the PNS problem

Operating units	Fixed charge	Proportional cost
O1	4	1
O2	3	1
O3	2	1
O4	2	0.5

Raw material	Constraint
E	≤ 10
Product	**Constraint**
A	≥ 4

Raw material	Price
E	0.8
F	1.6

PNS Studio: Optimal and 2nd best solution



Webpage: www.p-graph.com

Firefox | P-graph For: Process Synthesis; Process ... | www.p-graph.com | Google

P-graph www.p-graph.com is under continuous development.

P-graph For: Process Synthesis; Process Design; Process Optimization; Flowsheet Optimization; Process Systems Engineering; Process Network Synthesis (PNS)

- Home
- Introduction
- Literature Review
- Demonstration Programs
- Related Sites
- Editors
- Public Wiki

Mail to Webmaster

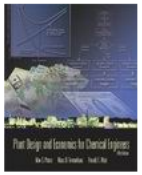
Welcome to www.p-graph.com !

2003 Computing in Engineering Award Winer: Professor Liang-Tseng Fan, Kansas State University
For broad and outstanding contributions to the analysis, synthesis, and control of process and material systems.

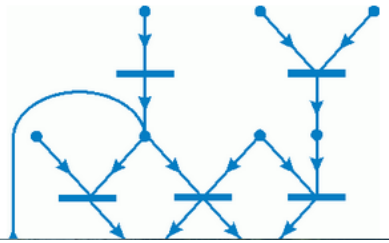
The CAST Award Lecture:
From Macroscopic World to Microscopic World through Mazes of Process Graphs and from Microscopic World to Mesoscopic World through Drunkards' Paths
by L. T. Fan

Download PowerPoint presentation.

Book chapter about the P-graph framework in a major chemical-engineering textbook



Web Site for Chapter 4
Flowsheet Synthesis and Development of Plant Design and Economics for Chemical Engineers
Fifth Edition
By Peters, Timmerhaus, and West



Literature

- Friedler, F.; Tarjan, K.; Huang, Y.; Fan, L. Graph-theoretic approach to process synthesis: polynomial algorithm for maximal structure generation. Comput. Chem. Eng. 1993, 17, 929.
- Friedler, F.; Varga, J.; Fan, L. Decision-mapping: a tool for consistent and complete decisions in process synthesis. Chem. Eng. Sci. 1995, 50, 1755.
- www.p-graph.com