ChBE 4746/6746

Homework 3 Solution

Fani Boukouvala, Pengfei Cheng, Zachary Kilwein

Update date: March 2, 2021

Problem 1 Flowsheet optimization with superstructure

(a) The superstructure is shown in Figure 1. Each flow is labeled with a red number under flow arrows, and all potential species in each flow is noted above flow arrows.

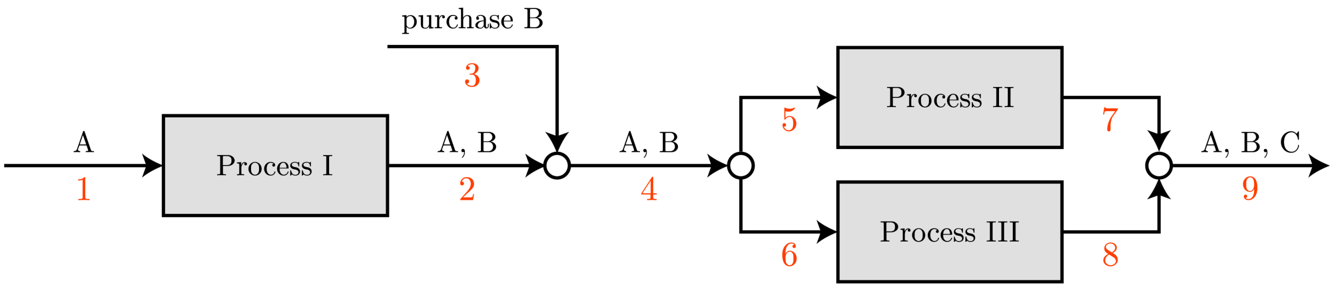

Figure 1: Problem 1 superstructure

(b) Define sets:

- $I = \{1, 2, \dots, 9\}$: flow set
- $J = \{\text{A}, \text{B}, \text{C}\}$: species set
- $K = \{\text{I}, \text{II}, \text{III}\}$: process set

Declare parameters:

- η_k: conversion rate of process k, $k \in K$
- c_j: raw material cost, \$/ton, $j \in J$
- c_k^{fc}: process fixed capital, \$/hr, $k \in K$
- c_k^{oc}: process operating cost, \$/ton, $k \in K$

Define variables:

- $f_{i,j}$: flowrate of species j in flow i, $i \in I, j \in J$
- y_k: whether process k is used, 1: yes, 0: no, $k \in K$
- x_{fc}: total fixed capital cost
- x_{oc}: total operating cost
- x_{c}: total raw material cost
- p: profit

(c) Define constraints

- Mass balance:

- Process I:

$$\eta_{\mathrm{I}} \cdot f_{1,\mathrm{A}} = f_{2,\mathrm{B}} \tag{1}$$

$$(1-\eta_{\mathrm{I}}) \cdot f_{1,\mathrm{A}} = f_{2,\mathrm{A}} \tag{2}$$

- mixing flow 2 and 3 into 4:

$$f_{2,\mathrm{A}} = f_{4,\mathrm{A}} \tag{3}$$

$$f_{2,\mathrm{B}} + f_{3,\mathrm{B}} = f_{4,\mathrm{B}} \tag{4}$$

- splitting flow 4 into 5 and 6:

$$f_{4,\mathrm{A}} = f_{5,\mathrm{A}} + f_{6,\mathrm{A}} \tag{5}$$

$$f_{4,\mathrm{B}} = f_{5,\mathrm{B}} + f_{6,\mathrm{B}} \tag{6}$$

Generally speaking, splitting should also include constraints to make sure that the ratio of A and B in flows 4, 5, 6 are identical:

$$\frac{f_{4,\mathrm{A}}}{f_{4,\mathrm{B}}} = \frac{f_{5,\mathrm{A}}}{f_{5,\mathrm{B}}} = \frac{f_{6,\mathrm{A}}}{f_{6,\mathrm{B}}}$$

(to avoid zero denominator, certain reformulation is necessary).
But in this case, as only one of Processes II and III will be chosen, there is only one nonzero outlet flow, so the ratio is automatically guaranteed, and we do not need the ratio constraints here.

- Process II:

$$f_{5,\mathrm{A}} = f_{7,\mathrm{A}} \tag{7}$$

$$(1-\eta_{\mathrm{II}}) \cdot f_{5,\mathrm{B}} = f_{7,\mathrm{B}} \tag{8}$$

$$\eta_{\mathrm{II}} \cdot f_{5,\mathrm{B}} = f_{7,\mathrm{C}} \tag{9}$$

- Process III:

$$f_{6,\mathrm{A}} = f_{8,\mathrm{A}} \tag{10}$$

$$(1-\eta_{\mathrm{III}}) \cdot f_{6,\mathrm{B}} = f_{8,\mathrm{B}} \tag{11}$$

$$\eta_{\mathrm{III}} \cdot f_{6,\mathrm{B}} = f_{8,\mathrm{C}} \tag{12}$$

- mixing flows 7 and 8 into 9[1]:

$$f_{7,\mathrm{A}} + f_{8,\mathrm{A}} = f_{9,\mathrm{A}} \tag{13}$$

$$f_{7,\mathrm{B}} + f_{8,\mathrm{B}} = f_{9,\mathrm{B}} \tag{14}$$

$$f_{7,\mathrm{C}} + f_{8,\mathrm{C}} = f_{9,\mathrm{C}} \tag{15}$$

- Binary constraint: only one of II and III will be used

$$y_{\mathrm{II}} + y_{\mathrm{III}} = 1 \tag{16}$$

[1]Some constraints are not necessary as the terms in them have no impact on the objective function. We here list all mass balance constraints just for completeness.

- Activation constraints: if process k is chosen, then its inlet flowrate should be within the given bounds; otherwise, its inlet flowrate should be 0
  - Process I: if process I is chosen, then $f_{1,\mathrm{A}}$ should not exceed 16; otherwise, it should be 0
$$f_{1,\mathrm{A}} \leq 16 y_{\mathrm{I}} \tag{17}$$
  - Process II: if process II is chosen, then $f_{5,\mathrm{B}}$ should not exceed $20/\eta_{\mathrm{II}}$[2]; otherwise, it should be 0
$$f_{5,\mathrm{B}} \leq 20/\eta_{\mathrm{II}} \cdot y_{\mathrm{II}} \tag{18}$$
  - Process III: if process III is chosen, then $f_{6,\mathrm{B}}$ should not exceed $20/\eta_{\mathrm{III}}$; otherwise, it should be 0
$$f_{6,\mathrm{B}} \leq 20/\eta_{\mathrm{III}} \cdot y_{\mathrm{III}} \tag{19}$$
- Total fixed capital:
$$x_{\mathrm{fc}} = \sum_{k \in K} c_k^{\mathrm{fc}} \cdot y_k \tag{20}$$
- Total operating cost:
$$x_{\mathrm{oc}} = c_{\mathrm{I}}^{\mathrm{oc}} f_{1,\mathrm{A}} + c_{\mathrm{II}}^{\mathrm{oc}} \left(f_{5,\mathrm{A}} + f_{5,\mathrm{B}}\right) + c_{\mathrm{III}}^{\mathrm{oc}} \left(f_{6,\mathrm{A}} + f_{6,\mathrm{B}}\right) \tag{21}$$
- Total raw material cost:
$$x_{\mathrm{c}} = c_{\mathrm{A}} \cdot f_{1,A} + c_{\mathrm{B}} \cdot f_{3,B} \tag{22}$$
- Profit:
$$p = 1800 f_{9,\mathrm{C}} \tag{23}$$
- Bounds
$$f_{i,j} \geq 0 \, \forall \, i \in I, j \in J, y_k \in \{0,1\} \, \forall \, k \in K, f_{1,\mathrm{A}} \leq 16, f_{9,\mathrm{C}} \leq 20 \tag{24}$$

(d) Define objective function:
$$\max p - x_{\mathrm{fc}} - x_{\mathrm{oc}} - x_{\mathrm{c}} \tag{25}$$

The corresponding model is an MILP problem. An alternative MINLP formulation is to replace the activation constraints with nonlinear constraints, for example: $f_{1,\mathrm{A}} \cdot y_{\mathrm{I}} \leq 16$. It functions in the same way as the linear constraint. *This formulation can receive full credits, but it is highly NOT recommended as it causes the model to be nonlinear and nonconvex.*

The optimal solution is as follows:

- Objective value (net profit): 2421.05 \$/hr
- $y_{\mathrm{I}} = 0, y_{\mathrm{II}} = 0, y_{\mathrm{III}} = 1$, so only Process III is chosen
- Raw material B purchase: 21.05 tons/hr
- Gross profit: 36000 \$/hr
- Total fixed capital: 2000.00 \$/hr, total operating cost: 11578.95 \$/hr, total raw material cost: 20000.00 \$/hr

The corresponding code is as follows:

[2] There is some flexibility here, numbers larger than this one will also lead to the correct answer as long as the final $f_{9,\mathrm{C}}$ is bounded; However it has an impact on the tightness of your relaxation in MILP.

```python
from pyomo.environ import *
from pyomo.opt import SolverStatus, TerminationCondition

m = ConcreteModel()

# sets
# stream set
m.I = Set(initialize=range(1, 10))
# chemical set
m.J = Set(initialize=['A', 'B', 'C'])
# process set
m.K = Set(initialize=['I', 'II', 'III'])

# parameters
# conversion rate
m.eta = Param(m.K, initialize={'I': 0.9, 'II': 0.82, 'III': 0.95})
# raw material cost: $/ton (C value does not matter, as it does not show up in the problem)
m.c = Param(m.J, initialize={'A': 550, 'B': 950, 'C': 0})
# process fixed capital: $/hr
m.fc = Param(m.K, initialize={'I': 1000, 'II': 1500, 'III': 2000})
# process operating cost: $/ton
m.oc = Param(m.K, initialize={'I': 250, 'II': 400, 'III': 550})

# variables
# flow
m.f = Var(m.I, m.J, initialize=0, within=NonNegativeReals)
# binary variables indicating if process k is used
m.y = Var(m.K, within=Binary)
# total fixed capital cost
m.x_fc = Var(within=NonNegativeReals)
# total operating cost
m.x_oc = Var(within=NonNegativeReals)
# total raw material cost
m.x_c = Var(within=NonNegativeReals)
# profit
m.p = Var(within=NonNegativeReals)
# bounds
m.f[1, 'A'].setub(16)
m.f[9, 'C'].setub(20)

# constraints
# ------------------------------------------------------------------------------
# mass balance
# Process I
m.mb_1 = Constraint(expr=m.f[2, 'B'] == m.eta['I'] * m.f[1, 'A'])
m.mb_2 = Constraint(expr=m.f[2, 'A'] == (1 - m.eta['I']) * m.f[1, 'A'])
# mixing flows 2 and 3 into 4
m.mb_3 = Constraint(expr=m.f[4, 'A'] == m.f[2, 'A'])
m.mb_4 = Constraint(expr=m.f[4, 'B'] == m.f[3, 'B'] + m.f[2, 'B'])
# splitting 4 into 5 and 6
m.mb_5 = Constraint(expr=m.f[4, 'A'] == m.f[5, 'A'] + m.f[6, 'A'])
m.mb_6 = Constraint(expr=m.f[4, 'B'] == m.f[5, 'B'] + m.f[6, 'B'])
# Process II
m.mb_7 = Constraint(expr=m.f[7, 'C'] == m.eta['II'] * m.f[5, 'B'])
```

```python
    m.mb_8 = Constraint(expr=m.f[7, 'B'] == (1 - m.eta['II']) * m.f[5, 'B'])
    m.mb_9 = Constraint(expr=m.f[7, 'A'] == m.f[5, 'A'])
    # Process III
    m.mb_10 = Constraint(expr=m.f[8, 'C'] == m.eta['III'] * m.f[6, 'B'])
    m.mb_11 = Constraint(expr=m.f[8, 'B'] == (1 - m.eta['III']) * m.f[6, 'B'])
    m.mb_12 = Constraint(expr=m.f[8, 'A'] == m.f[6, 'A'])
    # mixing flows 7 and 8 into 9
    m.mb_13 = Constraint(expr=m.f[9, 'A'] == m.f[7, 'A'] + m.f[8, 'A'])
    m.mb_14 = Constraint(expr=m.f[9, 'B'] == m.f[7, 'B'] + m.f[8, 'B'])
    m.mb_15 = Constraint(expr=m.f[9, 'C'] == m.f[7, 'C'] + m.f[8, 'C'])
    # ------------------------------------------------------------------------------
    # binary constraints
    # only one of II and III will be chosen
    m.bc = Constraint(expr=m.y['II'] + m.y['III'] == 1)
    # ------------------------------------------------------------------------------
    # activation constraints
    m.ac_1 = Constraint(expr=m.f[1, 'A'] <= 16 * m.y['I'])
    # m.ac_2 = Constraint(expr=m.f[5, 'A'] <= 16 * (1 - ) * m.y['II'])
    m.ac_3 = Constraint(expr=m.f[5, 'B'] <= 20 / m.eta['II'] * m.y['II'])
    # m.ac_4 = Constraint(expr=m.f[6, 'A'] <= 16 * m.y['III'])
    m.ac_5 = Constraint(expr=m.f[6, 'B'] <= 20 / m.eta['III'] * m.y['III'])
    # ------------------------------------------------------------------------------
    # cost and profit
    # total fixed capital cost
    m.fc_con = Constraint(expr=m.x_fc == summation(m.fc, m.y))
    # total operating cost
    m.oc_con = Constraint(expr=
                          m.x_oc ==
                          # Process I
                          m.oc['I'] * m.f[1, 'A'] +
                          # Process II
                          m.oc['II'] * (m.f[5, 'A'] + m.f[5, 'B']) +
                         # Process III
                          m.oc['III'] * (m.f[6, 'A'] + m.f[6, 'B']))
    # total raw material cost
    m.c_con = Constraint(expr=m.x_c == m.c['A'] * m.f[1, 'A'] + m.c['B'] * m.f[3, 'B'])
    m.p_con = Constraint(expr=m.p == 1800 * m.f[9, 'C'])

    # objective
    m.obj = Objective(expr=m.p - m.x_fc - m.x_oc - m.x_c, sense=maximize)

    # assign solvers and solve
    solver = SolverFactory('glpk')
    results = solver.solve(m)

    # print results
    # termination condition
    print(f"Termination condition: {results.solver.termination_condition}\n")
    # process selection
    print(f"Process selection:")
    for k in m.K:
        print(f"\ty{k:<3s} = {value(m.y[k]):n},", end="")
        if value(m.y[k]) == 1:
            print(f"\tProcess {k} selected")
        else:
            print(f"\tProcess {k} not selected")
    # process I
    if value(m.y['I']) == 1:
        print(f"\nRaw material A purchase: {value(m.f[1, 'A'])} tons/hr")
    # purchase B
    if value(m.f[3, 'B']) > 0:
        print(f"\nRaw material B purchase: {value(m.f[3, 'B']):.2f} tons/hr")
    # flowrate
    print(f"\nFlowrate (tons/hr):")
    print(f"\tA\tB\tC")
    for i in range(1, 10):
        if sum(value(m.f[i, j]) for j in m.J) > 0:
            print(f"{i}\t{value(m.f[i, 'A']):.2f}\t{value(m.f[i, 'B']):.2f}\t{value(m.f[i, 'C']):.2f}")
    # cost and profit
    print(f"\nfixed capital: {value(m.x_fc):.2f}$/hr\t"
          f"operating cost: {value(m.x_oc):.2f}$/hr\t"
          f"raw material cost: {value(m.x_c):.2f}$/hr")
    print(f"\ngorss profit: {value(m.p):.2f}$/hr")
    print(f"\nnet profit: {value(m.obj):.2f}$/hr")
```

Problem 2 Branch and bound

(a) The manual branch and bound can be done in the following way:

1) Relax the formulation in Problem 1 by changing the domain of y_k from $\{0, 1\}$ to $[0, 1]\ \forall k \in K$. This gives us Node 0. Solve Node 0 using an LP solver.
The corresponding code for this part is as follows. The formulation below is not entirely the same as the one in Problem 1 but should be identical.
Note:
   - If tighter bounds are used for the big-M constraints for $f_{5,\mathrm{B}}$ and $f_{6,\mathrm{B}}$ (e.g. 20/0.82 and 20/0.95 instead of 25, where 25 is a big-enough, but not carefully-chosen number), it is possible that we directly obtain the optimal solution at Node 0
   - If the problem is formulated as MINLP, then it should not require branching as there are no big-M constraints. This situation is acceptable, and can receive full credits. But we want to emphasize again that it is highly NOT recommended to formulate a problem as MINLP when we can formulate it with MILP.

```
from pyomo.environ import *

model = ConcreteModel()
model.i = Set(initialize=range(1,10))
model.j = Set(initialize=["A","B","C"])

model.n = Var(model.i,model.j,initialize=0,domain=NonNegativeReals)

# Relax Binary Variables For Node 0
model.y1=Var(within=NonNegativeReals, bounds=(0,1), initialize=0.5)
model.y2=Var(within=NonNegativeReals, bounds=(0,1), initialize=0.5)
model.y3=Var(within=NonNegativeReals, bounds=(0,1), initialize=0.5)

model.con1 = Constraint(expr = model.n[2,"B"]== 0.9*model.n[1,"A"])
model.con2 = Constraint(expr = model.n[2,"A"]== 0.1*model.n[1,"A"])
model.con3 = Constraint(expr = model.n[4,"A"]== model.n[2,"A"])
model.con4 = Constraint(expr = model.n[4,"B"]== model.n[2,"B"]+model.n[3,"B"])
model.con5 = Constraint(expr = model.n[5,"A"]+model.n[6,"A"]== model.n[4,"A"])
model.con6 = Constraint(expr = model.n[5,"B"]+model.n[6,"B"]== model.n[4,"B"])

model.con7 = Constraint(expr = model.n[7,"C"]== 0.82*model.n[5,"B"])
model.con8 = Constraint(expr = model.n[8,"C"]== 0.95*model.n[6,"B"])
model.con9 = Constraint(expr = model.n[9,"C"]== model.n[7,"C"]+model.n[8,"C"])

model.con10 = Constraint(expr = model.n[1,"A"]<=16*model.y1) # Maximum supply of A

# Activation Constraints are Looser Here to Show B&B Algorithm

model.con11 = Constraint(expr = model.n[5,"A"]<=16*model.y2)
model.con12 = Constraint(expr = model.n[5,"B"]<=25*model.y2)
model.con13 = Constraint(expr = model.n[6,"A"]<=16*model.y3)
model.con14 = Constraint(expr = model.n[6,"B"]<=25*model.y3)

model.con17 = Constraint(expr = model.n[7,"B"]== (1-0.82)*model.n[5,"B"])
model.con18 = Constraint(expr = model.n[8,"B"]== (1-0.95)*model.n[6,"B"])
model.con15 = Constraint(expr = model.n[9,"C"]<=20)

model.con16 = Constraint(expr = model.y2+model.y3==1)

model.obj = Objective(expr = 1800 * model.n[9,"C"] - 1000 * model.y1 - 1500 * model.y2 -
↪  2000 * model.y3
                      - 500 * model.n[1,"A"] - 950 * model.n[3,"B"]
                      - 250 * model.n[1,"A"] - 400 * (model.n[5,"A"] + model.n[5,"B"]) -
                      ↪  550 *
                      (model.n[6,"A"] + model.n[6,"B"]), sense=maximize)

solver = SolverFactory('glpk')
results = solver.solve(model)
print(f"Node 0 Objective Function: {value(model.obj):.2f}")
print(f"y1 = {value(model.y1):.2f}")
print(f"y2 = {value(model.y2):.2f}")
print(f"y3 = {value(model.y3):.2f}")
```

The solution is as follows:
- obj = 2540.00
- $y_1 = 1, y_2 = 0.16, y_3 = 0.84$

This provides an upper bound of the original problem.

2) We choose to branch y_2 first as it has the smallest index (it is also correct to branch y_3 first following a different branching rule).
We obtain Node 1 and 2 by fixing $y_2 = 1$ and $y_2 = 0$ respectively.
For Node 1, we can use the same code as Node 0 with an extra line:

```
model.y2.fix(1)
```

The solution is as follows:
- obj = 1613.17
- $y_1 = 1, y_2 = 1, y_3 = 0$

This provides a lower bound of the original problem, as all the binary variables are integer in the solution. So we prune this node.

3) For Node 2, we can use the same code as Node 0 with an extra line:

```
model.y2.fix(0)
```

The solution is as follows:
- obj = 2421.05
- $y_1 = 0, y_2 = 0, y_3 = 1$

This provides a lower bound of the original problem, as all the binary variables are integer in the solution. So we prune this node.

4) As all the nodes are pruned, there are no active nodes in the tree. So we terminate. The optimal solution corresponds to the node with the best lower bound, which is Node 2.
This solution is identical to our solution in Problem 1.

(b) The branch and bound tree is shown in Figure 2. Here we show full enumeration just for completeness. The actual tree only contains Node 0, 1, and 2 as we explained in part (a). Both nodes 1 and 2 are pruned because they provide lower bounds. Tighter activation constraints (con11-con14) allow for the same solution to be found at the root node.

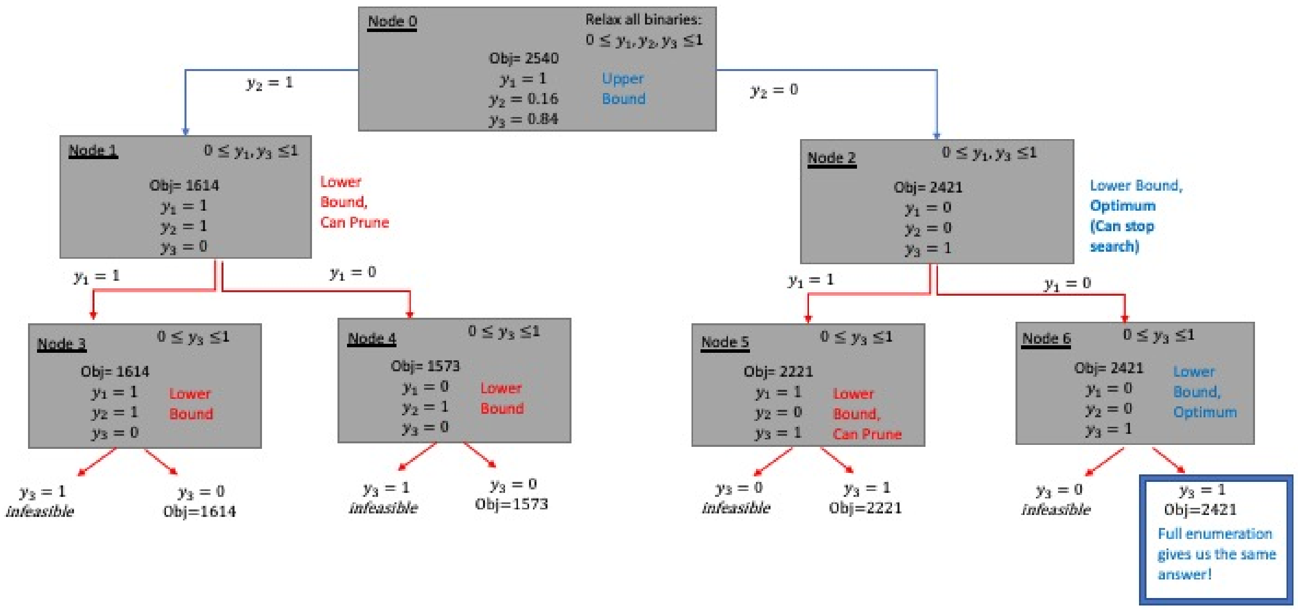

Figure 2: Branch and bound tree for Problem 2

(c) The optimal flowsheet is shown in Figure 3.

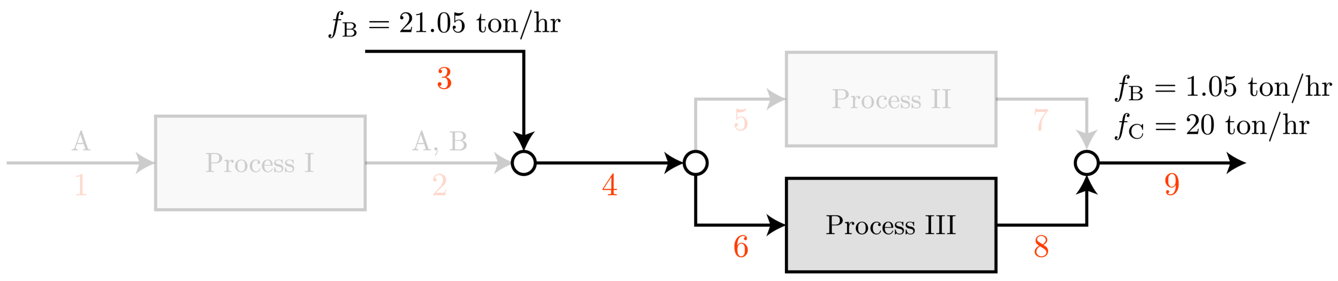

Figure 3: Optimal flowsheet for Problem 2