Optimization for Wastewater Piping and Treatment Cost in a Phenol Production Plant

Integrated Water Management Engineering (IWME 505), Dept. of Chemical Engineering
The University of British Columbia, *December 11, 2019*

Anuradha Ramachandran – 42197830
Srishti Birla – 14109565
Yami Patel - 41866369

Phenol production from cumene is a water intensive process, where water is used at different stages of production from washing cumene hydro peroxide to phenol formation. However, a substantial amount of wastewater is also generated from the same processes. Since phenol and the byproduct acetone are toxic chemicals further treatment of the process waste water is required to meet the environmental regulations. This presents an opportunity for water integration by recycling process water thereby limiting the consumption of freshwater. Here, two case scenarios are presented for this optimization problem. In the first case, a single objective linear programming model is developed to optimize the direct recycle network by minimizing process cost and the second scenario is formulated as a multi-objective optimization problem to include process cost as well as wastewater treatment costs.

***Index Terms* — Multi objective optimization, wastewater recycling, process flow, phenol production, piping cost, treatment cost, minimizing cost**

NOMENCLATURE

PFD – Process Flow Diagram, COD – Chemical Oxygen Demand, ThOD – Theoretical Oxygen Demand, CHP – Cumene hydroperoxide

I. INTRODUCTION

Industrial waste is released into the environment every day and can quickly and irreversibly damage the environment, affecting human health. Therefore, environmental protection is one of our top priorities worldwide. Strict environmental regulations have been introduced to control and reduce pollutants released into the environment. These restrictions are severely impacted by the processing industry as they contribute significantly to the technical, operational and economic issues of the process. The goal of protecting natural resources and reducing industrial pollution is to discover new ways to optimize process performance and reduce waste disposal.

Process control plays an important role in shaping pollution changes and reducing design. In particular, the engineering process known as process integration forms a complex system for process design and operation. Various optimization processes have been developed to integrate materials, resources and wastes through integrated processes. Today, most methods are usually based on processes or environmental constraints, but they are not. By streamlining processes, waste management can be simplified, but it can increase management costs and environmental performance. Pollution prevention is based on pollution control. By integrating stream processes and waste quality, this publication aims to address environmental issues together. There are limitations and functionality. A systematic design methodology has been developed to provide the following benefits:

- Minimization of the total cost of fresh resources and waste treatment while satisfying process and environmental constraints.

- Determination of a trade-offs between the cost of fresh resources versus the cost of environmental compliance and pollution prevention.

- Development of implementation projects needed to achieve the target at minimum cost.

The environmental regulations including chemical oxygen demand (COD), toxicity, and pH were taken into consideration in addition to process constraints. In order to demonstrate the applicability of the developed approach, a case study is solved to address resource conservation and pollution prevention of a phenol process.

II. Process Description

The figure II-1, schematically shows a process flowchart for the preparation of phenol from cumene hydroperoxide (CHP). Cumene is chosen as a raw material. First, the feed (ie, cumene) is introduced into the reactor along with air and Na2CO3, serving as a buffer solution. In the reactor, cumene is oxidized by atmospheric oxygen in cumene hydroperoxide (CHP). The CHP and cumene mixture is then sent to a wash to remove excess buffer and water-soluble materials. The power of the washer is then directed to a concentration unit to increase the low concentration of CHP to 80% by weight or more. The concentrated stream of maleic hydrogen hydroxide is then passed to cleavage units where CHP is decomposed in the presence of sulfuric acid to form phenol and acetone. The resulting cleavage stream is neutralized with a small amount of sodium hydroxide and separated into two phases (ie the organic phase and water). The aqueous phase is taken to wastewater treatment. Meanwhile, the organic phase, which is essentially a mixture of phenol, acetone and cumene, is treated in a purification plant to remove excess base and finally part of the distillation columns, where they are fractionated into pure phenol products and acetone.

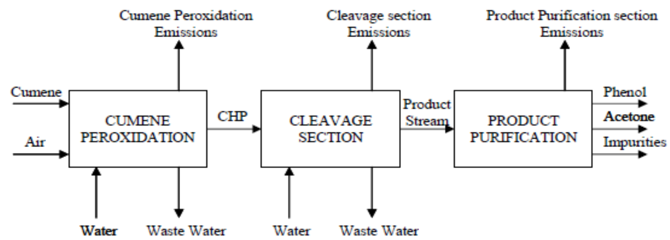

Figure II-2 Simplified process flow diagram for the manufacture of acetone and phenol from cumene.

III. Data Extraction

In this study, water was selected as a new function for process integration. As a result, the operating data, property values and operating costs of water consumption and dose processing units are immediately analyzed. Figure III-1 shows a detailed flow chart of the cumene oxidation and slice cleavage process. In the process description and in Figure III-1, the process units, process flows and new functions that are linked to this case study are summarized as follows:

Process sinks:

1. Waterwash cumene peroxidation section (Wash101)
2. Neutralizer (R104)
3. Waterwash cleavage section (Wash102)

Process sources:

1. Stream 8 from Wash101
2. Stream 22 from Decanter (D101)
3. Stream 25 from Wash102

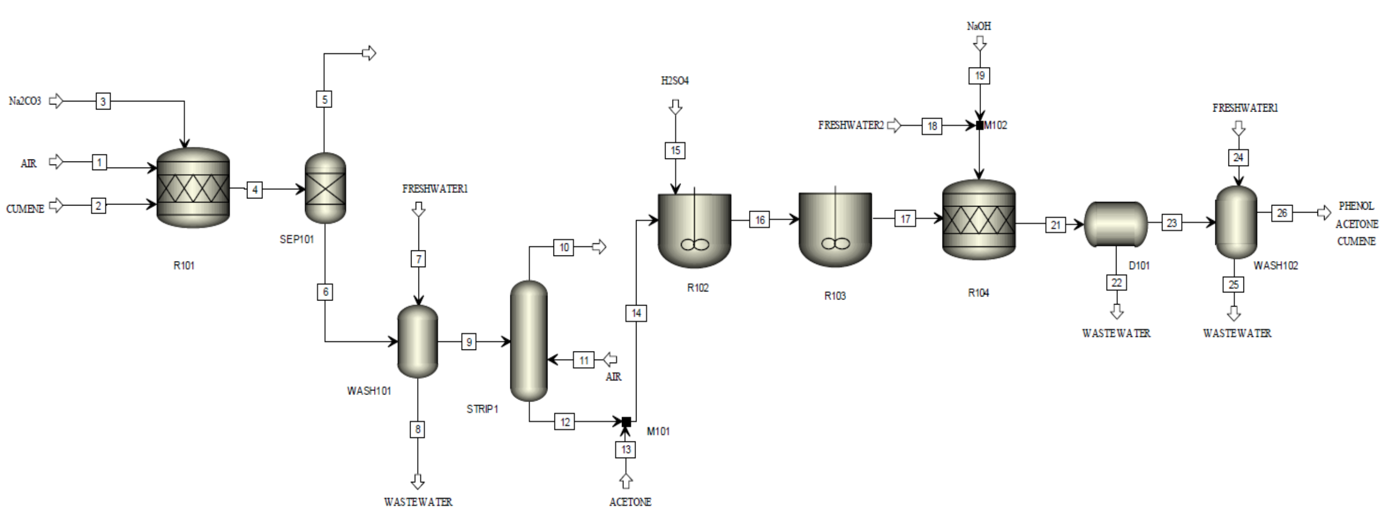

Figure II-1 Process Flow diagram - cumene peroxidation and cleavage sections

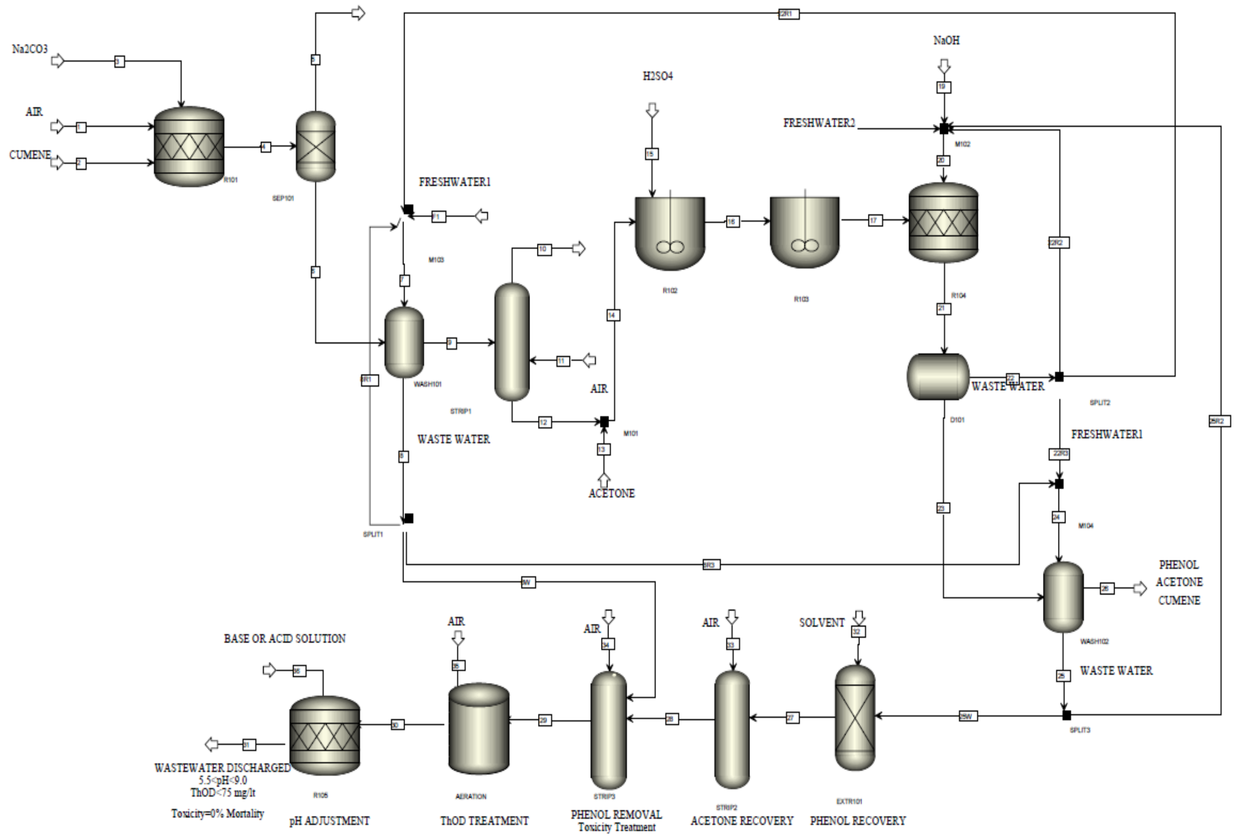

Figure III-1 Phenol PFD after Optimization with treatment process

Fresh water sources:

1. Freshwater 1: 0 impurity concentration
2. Freshwater 2: 0.012 impurity concentration (mass fraction)

The figure 3.2 follows the same link but with the inclusion of the treatment facility. It starts with phenol recovery unit where phenol is recovered the output by recycling the same and thus decreasing the amount the wastewater according to the environmental regulations. Then comes the acetone recovery having the same purpose as phenol recovery. Followed by Thod (Theoretical oxygen demand) treatment, where the COD of the wastewater is improved. Last is the pH treatment which helps to bring the pH of the wastewater to acceptable amount of around 7.

In order to analyze the effect of adding a wastewater treatment facility, two optimization programs are examined. The first model examines a single objective optimization problem with only process constraints and the second model examines a multi-objective optimization problem with both process as well as wastewater treatment considerations.

IV. PIPING COST OPTIMIZATION

Single objective optimization

This model formulation is based on direct material recovery i.e., the waste produced by the process can be discharged without any further treatment into the environment. The objective of this formulation is to identify the recycle stream, the freshwater stream and waste stream that minimizes the cost of piping recycle materials between the processes and the cost of consumption of freshwater resources. From the pinch analysis given in the literature [1], three process source streams and three sink streams have been identified. They are given in tables IV-1 and IV-2.

Table IV-1 Sources

Sources	*Nomenclature*	*Stream number*
Washer (Source 1)	WASH 101	Stream 8
Decanter (Source 2)	D101	Stream 22

Washer (Source 3)	WASH 101	Stream 25

Table IV-2 Sinks

Sinks	*Nomenclature*
Water wash (cumene peroxidation section)	WASH 101
Neutralizer	R104
Water wash (cleavage section)	WASH 102

In addition to three process sources, two freshwater sources (FRESH 1 & FRESH 2) are also considered in the product formulation. The graphical representation of the problem is as shown in Figure IV-1.

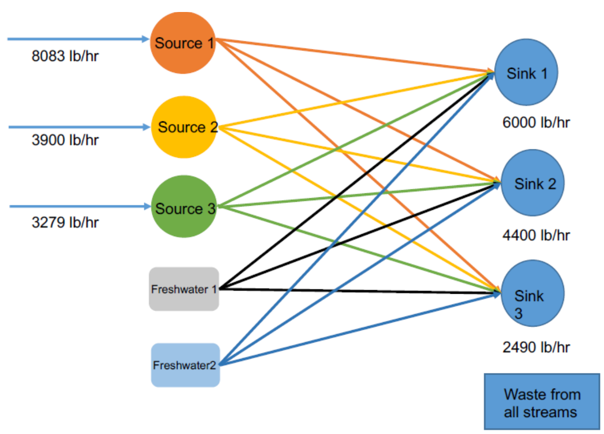

Figure IV-1 Single objective optimization problem representation

Objective function

The decision variables in the problem statement include all recycle network connections between the sources and sinks, input stream from the freshwater sources to the sinks as well as the process waste streams. Total Decision variables = 18

- Recycle streams = dij, i=1…3, j=1…3
- Freshwater streams = Fij, i=1..2, j=1..3
- Waste stream = Wi, i=1..3

The objective function in this model is the minimization of cost both piping cost as well as the cost of freshwater consumption.

Piping cost coefficient for each recycle stream and freshwater stream is given in table IV-3.

Table IV-3 Piping cost coefficients

Sources	*Source 1*	*Source 2*	*Source 3*	*Fresh 1*	*Fresh 2*
Sinks					
Sink 1	5	2	3	4.5	2.5
Sink 2	3.5	1	5	3	1
Sink 3	2	4	2	3.5	1.5

Freshwater cost coefficients are 0.0006 \$/lb and 0.0004\$/lb for Fresh 1 & 2 respectively.

Objective function:

$$Min \sum_{i=1}^{3}\sum_{j=1}^{3} d_{ij} * Pipecost_{ij} + \sum_{i=1}^{2}\sum_{j=1}^{3} F_{ij} * Pipecost_{ij} + F_{ij} * Freshcost_i$$

Constraints

The above-mentioned objective function is subjected to two sets of constraints namely, mass balance and impurity constraint.

From the conservation of mass principle, $Mass\ in = Mass\ out + Mass\ Accumulation - Mass\ generation$

In this system, there is no accumulation or generation of mass in these process units. Hence the mass balance constraint for each source becomes,

$$\sum_{i=1}^{3}\sum_{j=1}^{3} d_{ij} + \sum_{i=1}^{3} W_i = \sum_{i=1}^{3} Mass\ flowrate\ from\ the\ source$$

Moreover, the process water from the source and the freshwater sources has impurities and the process water to the sinks have to meet the required purity for the production of phenol. The impurity concentration in all sources and the sink stream are in Table IV-4 and IV-5.

Table IV-4 Impurity concentration in Sources

Source streams	*Source 1*	*Source 2*	*Source 3*	*Fresh 1*	*Fresh 2*
Impurity fraction	0.016	0.024	0.22	0	0.012

Table IV-5 Allowable impurity in Sink

Sink streams	*Sink 1*	*Sink 2*	*Sink 3*
Allowable impurity fraction	0.013	0.013	0.1

The impurity constraint can be stated as below

$$\sum_{i=1}^{3}\sum_{j=1}^{3} d_j * Source\ impurity_i + \sum_{i=1}^{2}\sum_{j=1}^{3} f_{ij} * Fresh\ impurity_i = \sum_{j=1}^{3} Allowed\ Sink\ impurity_j$$

Optimization Results

This linear programming problem with 18 variables, 9 equality constraints and 18 non-negativity constraints was solved in JuMP Julia and the optimized recycle network is given in Figure IV-2.

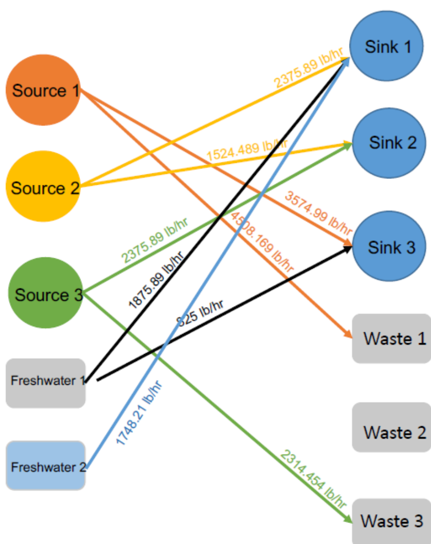

Figure IV-2 Optimized Recycle Network

With this optimized recycle network, the optimal cost was found to be 52511.9518 $/day.

Sensitivity analysis

The shadow prices for the constraints were found in JuMP Julia and are in Table IV-6.

Table IV-6 Sensitivity analysis - Shadow Prices

Constraints	*Shadow Prices*
Constraint 1	0
Constraint 2	-0.1
Constraint 3	0
Constraint 4	-9.3
Constraint 5	-0.62
Constraint 6	-8.30
Constraint 7	-300
Constraint 8	-19.90
Constraint 9	-393.75

It can be inferred that, due to the equality constraints used in the problem formulation, there is no room for flexibility in the constraint coefficients and the right-hand side values.

V. Piping Cost and Treatment Cost Optimization

Multi objective optimization

The first objective remains the same as the case for single objective i.e., to reduce piping and freshwater cost. Along with reduction in cost for transport of recycled and fresh process water, as a plant manager, another consideration will be reduction of cost of treatment of water needed to be disposed from the plant.

The total treatment cost is a function of cost due to phenol recovery, acetone recovery, theoretical oxygen demand, acid and base neutralization.

Theoretically, objective 2 can be broken down as follows:

Total Treatment Cost = Phenol Recovery Cost

$$\left[f\left(\begin{matrix}\text{fraction of phenol in inlet of waste stream,}\\ \text{efficiency of phenol recovery column}\end{matrix}\right)\right]+$$

Acetone Recovery Cost

$$\left[f\left(\begin{matrix}\text{fraction of acetone in inlet of waste stream,}\\ \text{efficiency of acetone recovery column}\end{matrix}\right)\right]+$$

Theoretical Oxygen Demand (ThOD) Cost

$[f(\text{Phenol discharged} \leq 0.000011\ \text{lb phenol/lb wastewater})]$

+Acid and Base Neutralization Cost $[f(5.5 \leq pH \leq 8.5)]$

Phenol Recovery

Phenol is recovered using an extraction column.

Flowrate of phenol removed from waste stream ‘i’ is calculated as a multiple of efficiency of phenol column, concentration of phenol in waste stream i, flowrate of waste stream i and a binary number signifying if the phenol recovery is required.

Phenol remaining = (1- efficiency of column) × sum of waste in streams 1 to i × concentration of phenol in waste streams.

From this, we can calculate phenol waste discharged after recovery of phenol.

Thus, Cost of phenol recovery = total phenol recovered × cost for removal of phenol.

Acetone Recovery

Similar to phenol recovery, acetone is recovered using an extraction column. Similarly,

Cost of acetone recovery = total acetone recovered multiplied by cost for removal of acetone.

Other considerations for recovery costs include environmental regulations and permits, allowable release and default costs.

Theoretical Oxygen Demand Treatment

An aeration basin is used to remove toxicity and increase oxygen before removal of wastewater. Concentration of organic compounds; phenol, acetone, Isopropyl alcohol, hydroperoxide, acetophenone, dimethyl and cumene, in the wastewater stream is calculated. Based on the concentration, total theoretical oxygen demand in waste streams can be calculated. The oxygen to be diffused is a function of total ThOD of waste streams, regulation imposed on ThOD and waste remaining after toxicity treatment.

The cost for ThOD Treatment is thus a function of amount of air required and cost for the same.

The ThOD regulation states that ThOD <= 75.

pH Treatment

The wastewater to be discharged must have a neutral pH. Since the impurity compounds are either highly acidic or basic, they must be treated with bases or acids respectively in order to bring a neutral pH. According to guidelines, some flexibility for pH is allowed. The pH is allowed to be in the range of 5.5 – 8.5.

The pH of the waste stream requires calculation of H_3O^+ and OH^- in the streams in order to calculate the amount of neutralization reagent required. Based on the pH of the inlet streams, a binary function is used to determine if the treatment is required.

The total acid treatment cost is amount of base required*cost of base. Similarly, total base treatment cost is amount of acid required*cost of acid.

Thus, Treatment cost is an implicit function of amount of wastewater streams 1, 2 and 3.

Objective Function 2

A limitation for using JuMP optimizer in Julia is that an implicit objective function for treatment cost cannot be used. Hence, a multiplier 'm' must be calculated in order to make the objective function explicit in wastewater streams. Such that,

$$Treatment\ Cost = m \times (waste\ stream\ 1 + waste\ stream\ 2 + wastestream\ 3)$$

In order to calculate 'm', the Treatment Cost code was run for different amount of waste stream inlets and plotted. The $R^2 = 1$, treatment cost is a linear function of total amount of wastewater.

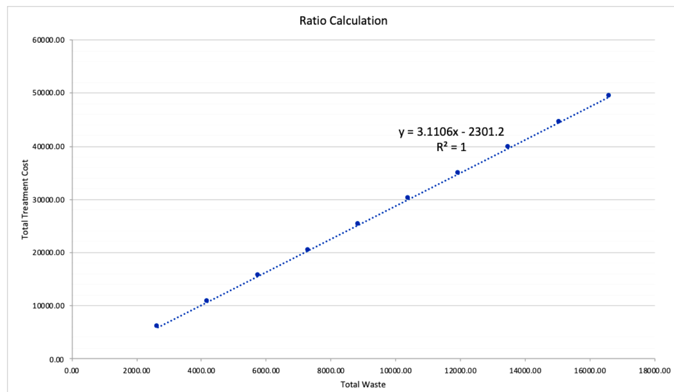

Figure V-1 Calculation of multiplier 'm' for 2nd Objective

The graph gives the slope which can be used as a multiplier for the second objective. The slope calculated from graph is 3.1106.

Thus, the objective function 2 can be represented as

$$Minimize\ Treatment\ Cost = 3.1106 \times (waste\ stream\ 1 + waste\ stream\ 2 + waste\ stream\ 3)$$

The optimization is solved using weighting method.

The results obtained provides three values for total cost. The solutions are represented in Table V-1 and Figure V-2.

Table V-1 Solution for multi objective optimization

Objective 1	*Objective 2*	*Total Cost*
52511.95	21222.45	73734.41
53167.53	17143.94	70311.48
54979.17	13829.08	**68808.25**

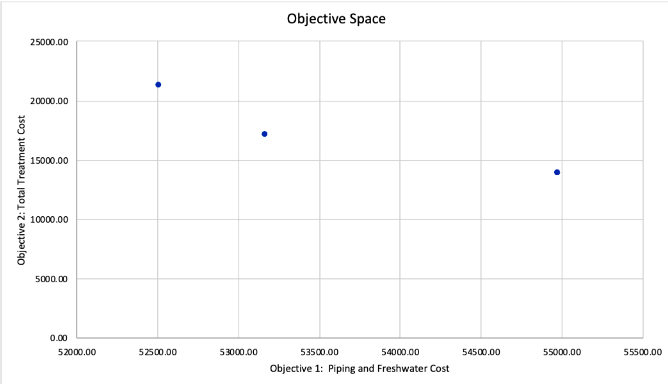

Figure V-2 Objective Space for multi objective optimization

As a plant manager, the objectives solved would give the value of total cost to be $68,808.25, since the objective is to minimize total cost.

VI. Conclusion

The piping cost for recycling wastewater with additional freshwater for phenol production plant is $52511.95/day.

Treatment cost is included in order to ensure the environmental regulations are met. This leads to a total cost of $68808.25/day, with minimal treatment cost.

Reference

1	Hortua, Ana Carolina, Chemical Process optimization and pollution prevention via Mass and Property Integration, May 2007

APPENDIX

Code and Results for Single and Multi-objective Optimization in JuMP Julia

In [2]:

```
using Pkg
Pkg.add("JuMP")
Pkg.add("Clp") #the free solver
Pkg.add("Plots")
```

```
  Updating registry at `/srv/julia/pkg/registries/Gene
ral`
  Updating git-repo `https://github.com/JuliaRegistrie
s/General.git`
[1mFetching: [========================================
>]  99.9 %0.0 %]  2.7 %Fetching: [=====>
]  11.4 %>                              ]  23.5 %
]  38.3 %                 ]  52.9 %Fetching:
[========================>                ]  55.5 %>
]  67.6 %Fetching: [=============================>
]  70.4 %>      ]  82.6 %Fetching:
[=====================================>   ]  90.5 %Fet
ching: [=======================================> ]  95
.5 % Resolving package versions...
 Installed NaNMath ───────────────────── v0.3.3
 Installed AxisAlgorithms ────────────── v1.0.0
 Installed Juno ──────────────────────── v0.7.2
 Installed Parsers ───────────────────── v0.3.10
 Installed Colors ────────────────────── v0.9.6
```

In [3]:

```
using JuMP, Clp, Plots
```

```
┌ Info: Precompiling JuMP [4076af6c-e467-56ae-b986-b46
6b2749572]
└ @ Base loading.jl:1186
┌ Info: Precompiling Clp [e2554f3b-3117-50c0-817c-e040
a3ddf72d]
└ @ Base loading.jl:1186
┌ Info: Precompiling Plots [91a5bcdd-55d7-5caf-9e0b-52
0d859cae80]
└ @ Base loading.jl:1186
```

In [4]:

```
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, sum((PipCost[v]+FWCost[v])*Var[v] for
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
```

```
Coin0506I Presolve 9 (0) rows, 15 (-3) columns and 36
(-3) elements
Clp0006I 0  Obj 3556.6408 Primal inf 16610.429 (6)
Clp0006I 11  Obj 52511.952
Clp0000I Optimal - objective value 52511.952
Coin0511I After Postsolve, objective 52511.952, infeas
ibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 52511.9518 - 11 iterations
time 0.002, Presole 0.00
```

Out[4]:

```
52511.951795918365
```

In [5]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[5]:

```
9-element Array{Float64,1}:
    0.0
   -0.09999999999999766
    0.0
   -9.3
   -0.6224489795918341
   -8.3
 -300.0000000000001
  -19.897959183673482
 -393.75000000000006
```

In [ ]:

```
```

In [1]:

```julia
using Pkg
Pkg.add("JuMP")
Pkg.add("Clp") #the free solver
Pkg.add("Plots")
```

```
  Updating registry at `/srv/julia/pkg/registries/Gene
ral`
  Updating git-repo `https://github.com/JuliaRegistrie
s/General.git`
[1mFetching: [========================================
>]  99.9 %0.0 %>                                ]  26.1 %
Fetching: [=============>                           ]
30.1 %>                       ]  42.1 %]  43.7 %======
=================>                        ]  55.8 %Fetching:
[=======================>                 ]  57.1 %>
]  69.1 %>     ]  85.0 %Fetching:
[=======================================>  ]  93.6 %Fet
ching: [=========================================> ]  95
.2 % Resolving package versions...
 Installed NaNMath ───────────────────── v0.3.3
 Installed AxisAlgorithms ────────────── v1.0.0
 Installed Juno ──────────────────────── v0.7.2
 Installed Parsers ───────────────────── v0.3.10
 Installed Colors ────────────────────── v0.9.6
 Installed JuMP ──────────────────────── v0.20.1
```

In [2]:

```julia
using JuMP, Clp, Plots
```

```
┌ Info: Precompiling JuMP [4076af6c-e467-56ae-b986-b46
6b2749572]
└ @ Base loading.jl:1186
┌ Info: Precompiling Clp [e2554f3b-3117-50c0-817c-e040
a3ddf72d]
└ @ Base loading.jl:1186
┌ Info: Precompiling Plots [91a5bcdd-55d7-5caf-9e0b-52
0d859cae80]
└ @ Base loading.jl:1186
```

In [31]:

```
wt = 1;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 15 (-3) columns and 36
(-3) elements
Clp0006I 0  Obj 3556.6408 Primal inf 16610.429 (6)
Clp0006I 11  Obj 52511.952
Clp0000I Optimal - objective value 52511.952
Coin0511I After Postsolve, objective 52511.952, infeas
ibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 52511.9518 - 11 iterations
time 0.002, Presolve 0.00
```

Out[31]:

```
21222.45357958367
```

In [32]:

```
has_duals(PhenolLP)
```

Out[32]:

```
true
```

In [33]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[33]:

```
9-element Array{Float64,1}:
     0.0
    -0.09999999999999766
     0.0
    -9.3
    -0.6224489795918341
    -8.3
  -300.0000000000001
   -19.897959183673482
  -393.75000000000006
```

In [34]:

```
wt = 0.9;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 12  Obj 49802.184
Clp0000I Optimal - objective value 49802.184
Clp0032I Optimal objective 49802.18398 - 12 iterations
time 0.002
```

Out[34]:

```
21222.453579583667
```

In [35]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[35]:

```
9-element Array{Float64,1}:
   -0.37249999999999994
   -0.09000000000000102
   -0.3724999999999999
   -8.370000000000001
   -0.6058163265306132
   -7.470000000000001
 -269.99999999999994
  -16.007653061224488
 -377.65625000000006
```

In [36]:

```
wt = 0.8;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 11  Obj 46640.064
Clp0000I Optimal - objective value 46640.064
Clp0032I Optimal objective 46640.06435 - 11 iterations
time 0.002
```

Out[36]:

```
17143.944682808164
```

In [37]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[37]:

```
9-element Array{Float64,1}:
   -0.7449999999999999
   -0.4375000000000144
   -0.7449999999999998
   -7.4399999999999995
   -0.008316326530610653
   -6.640000000000001
 -261.5625
  -14.757653061224497
 -361.56250000000006
```

In [38]:

```
wt = 0.7;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 11  Obj 43376.329
Clp0000I Optimal - objective value 43376.329
Clp0032I Optimal objective 43376.32945 - 11 iterations
time 0.002
```

Out[38]:

```
17143.944682808164
```

In [39]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[39]:

```
9-element Array{Float64,1}:
   -1.1175000000000002
   -1.0812499999999996
   -1.1175000000000002
   -6.51
   -0.7196683673469381
   -5.81
 -257.96874999999994
  -14.100765306122446
 -345.46875000000006
```

In [40]:

```
wt = 0.6;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 13  Obj 39611.731
Clp0000I Optimal - objective value 39611.731
Clp0032I Optimal objective 39611.73054 - 13 iterations
time 0.002
```

Out[40]:

```
13829.0774846
```

In [41]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[41]:

```
9-element Array{Float64,1}:
   -1.3333333333333328
   -1.4900000000000004
   -1.4900000000000002
   -5.58
   -1.1838775510204085
   -4.98
 -244.58333333333337
  -12.244897959183675
 -319.5833333333333
```

In [59]:

```
wt = 0.5;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 12  Obj 35769.871
Clp0000I Optimal - objective value 35769.871
Clp0032I Optimal objective 35769.8707 - 12 iterations
time 0.002
```

Out[59]:

```
13829.0774846
```

In [60]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[60]:

```
9-element Array{Float64,1}:
   -1.525000000000001
   -1.8625000000000005
   -1.8625
   -4.65
   -1.6073979591836738
   -4.15
 -229.68750000000009
  -10.204081632653063
 -292.1875000000001
```

In [14]:

```
wt = 0.4;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 12  Obj 31928.011
Clp0000I Optimal - objective value 31928.011
Clp0032I Optimal objective 31928.01085 - 12 iterations
time 0.002
```

Out[14]:

```
54979.16991632653
```

In [43]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[43]:

```
9-element Array{Float64,1}:
   -1.525000000000001
   -1.8625000000000005
   -1.8625
   -4.65
   -1.6073979591836738
   -4.15
 -229.68750000000009
  -10.204081632653063
 -292.1875000000001
```

In [54]:

```
wt = 0.3;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 13  Obj 28086.151
Clp0000I Optimal - objective value 28086.151
Clp0032I Optimal objective 28086.15101 - 13 iterations
time 0.002
```

Out[54]:

```
13829.0774846
```

In [56]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[56]:

```
9-element Array{Float64,1}:
   -1.9083333333333334
   -2.6075000000000004
   -2.6075
   -2.79
   -2.4544387755102046
   -2.49
 -199.89583333333337
   -6.122448979591842
 -237.3958333333334
```

In [16]:

```
wt = 0.2;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 12  Obj 24244.291
Clp0000I Optimal - objective value 24244.291
Clp0032I Optimal objective 24244.29116 - 12 iterations
time 0.002
```

Out[16]:

```
54979.16991632653
```

In [45]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[45]:

```
9-element Array{Float64,1}:
   -1.9083333333333334
   -2.6075000000000004
   -2.6075
   -2.79
   -2.4544387755102046
   -2.49
 -199.89583333333337
   -6.122448979591842
 -237.3958333333334
```

In [46]:

```
wt = 0.1;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 18 (0) columns and 39 (
0) elements
Clp0006I 0  Obj 0 Primal inf 34348.389 (9)
Clp0006I 12  Obj 20402.431
Clp0000I Optimal - objective value 20402.431
Clp0032I Optimal objective 20402.43132 - 12 iterations
time 0.002
```

Out[46]:

```
13829.0774846
```

In [47]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[47]:

```
9-element Array{Float64,1}:
   -2.291666666666674
   -3.352500000000005
   -3.3524999999999996
   -0.9299999999999999
   -3.3014795918367352
   -0.8300000000000001
 -170.10416666666669
   -2.040816326530616
 -182.60416666666674
```

In [49]:

```
wt = 0;
    N = 9 #No. of Constraints
    PipCost = [5,3.5,2,2,1,4,3,5,2,4.5,3,3.5,2.5,1,1.5,0,0,0] #Pipi
    FWCost= [0,0,0,0,0,0,0,0,0,4.8,4.8,4.8,3.2,3.2,3.2,0,0,0] #Fres
    R = [8083.169, 3900.38,3279.965, 6000,2490,4400,78,249,57.2] #R
    T = [[1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]; [0 0 0 1 1 1 0 0 0
        [0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0];[0 0 1 0 0 1 0 0 1 0
    # T = Coefficient Matrix
    V = 18 #No. of Variables

    PhenolLP = Model(with_optimizer(Clp.Optimizer))
    @variable(PhenolLP, Var[i=1:18]>=0)
    Z1 = sum((PipCost[v]+FWCost[v])*Var[v] for v in 1:V)
    Z2 = (3.1106)*(Var[16]+Var[17]+Var[18])

#objective #Minimize total cost = (PipCost[v] + FWCost[v])*X[v] for
    @objective(PhenolLP, Min, wt*sum((PipCost[v]+FWCost[v])*Var[v]
    @constraint(PhenolLP, CONSTRA, T*Var .== R) #Piping

    JuMP.optimize!(PhenolLP)
    JuMP.value.(Var)
JuMP.value(Z1)
JuMP.value(Z2)
```

```
Coin0506I Presolve 9 (0) rows, 15 (-3) columns and 36
(-3) elements
Clp0006I 0  Obj 5712.2924 Primal inf 15183.511 (6)
Clp0006I 14  Obj 16560.571
Clp0000I Optimal - objective value 16560.571
Coin0511I After Postsolve, objective 16560.571, infeas
ibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 16560.57148 - 14 iterations
time 0.002, Presolve 0.00
```

Out[49]:

```
13829.0774846
```

In [50]:

```
JuMP.shadow_price.(CONSTRA)
```

Out[50]:

```
9-element Array{Float64,1}:
   -2.483333333333334
   -3.7250000000000001
   -3.725
    0.0
   -3.7250000000000005
    0.0
 -155.20833333333337
    0.0
 -155.2083333333334
```

In [48]:

```
W = [0.0 0.0 0.0]
r1 = 1000;
r2 = 50;
r3 = 100;

for y = 1:10
    W[1] = 1000 + r1;
    W[2] = 0 + r2;
    W[3] = 500 + r3;
    WasteStreamCalc = W[1] + W[2] + W[3];
    IP = [0.0, 0.0, 0.0]
    ini = 0.0
    WRecPhenol = [0.0, 0.0, 0.0]
    WARPhenol = [0.0, 0.0, 0.0]
    CostPhenol = 0
effPhenol = 0.93
PhenolRecCost = 0.65*8000



Zphenol = [0.000017 0.013 0.024]

#Phenol in Waste
RecPhenol = 500*10^-6
Wasteafterrecovery = [0.0 0.0 0.0]

    for i = 1:1:3
```

```
        if Zphenol[i] >= RecPhenol
            IP[i] = 1.0
        else
            IP[i] = 0.0
        end
        WRecPhenol[i] = effPhenol * W[i]*Zphenol[i]*IP[i]
        Wasteafterrecovery[i] = W[i] - WRecPhenol[i]
        if IP[i] == 1
            WARPhenol[i] = (1-effPhenol)*W[i]*Zphenol[i]
        else
            WARPhenol[i] = W[i]*Zphenol[i]
        end
        CostPhenol = ini + WRecPhenol[i]*PhenolRecCost
        ini = CostPhenol
    end


iniA = 0
IA = [0.0 0.0 0.0]
WRecAcetone = [0.0 0.0 0.0]
WARAcetone = [0.0 0.0 0.0]
CostAcetone = 0;
ZAcetone = [0 0.01 0.028] #Acetone in Waste Streams
effAcetone = 0.98
RecAcetone = 500*(10^-6)
AcetoneRecCost = 0.033*8000

for i = 1:1:length(ZAcetone)
    if ZAcetone[i] >= RecAcetone
        IA[i] = 1
        else IA[i] = 0;
    end

    WRecAcetone[i] = effAcetone*W[i]*ZAcetone[i]*IA[i]; #WRecAceton
    WARAcetone[i] = Wasteafterrecovery[i] - WRecAcetone[i] #final w
    CostAcetone = iniA + (WRecAcetone[i]*AcetoneRecCost)
    iniA = CostAcetone
end




TotalRecCost = CostPhenol + CostAcetone #total recovery cost

#Toxicity Treatment
#Zdischarged Phenol <= 0.00000011 lb phenol/lb waste
```

```
i = 1
    totalphenolremaining =  (WARPhenol[i] + WARPhenol[i+1] + WARPhe
    TotafterAcRec =(WARAcetone[i] + WARAcetone[i+1] + WARAcetone[i+

WasteaftTox = 0.0;
ywasteIsoProp = 0.0;
ywasteHydroA = 0.0;
ywastePhenone = 0.0;
ywasteDimethyl = 0.0;
ywasteCumene = 0.0;
ywastePhenol = 0.0;
ywasteAcetone = 0.0;

ZMeanPhenol = totalphenolremaining/TotafterAcRec #Phenol conc. ente

ZdischargePhenol = 0.0000011


if ZMeanPhenol > ZdischargePhenol
    ZPhenolremoved = ZMeanPhenol - ZdischargePhenol #Phenol conc. n
    WPhenolremoved = ZMeanPhenol*TotafterAcRec #load of phenol remo
    WasteaftTox = TotafterAcRec - WPhenolremoved #Waste after toxic
    Phenolrem = ZdischargePhenol*WasteaftTox #phenol remaining afte
    PhenolToxCost = WPhenolremoved*0.164*8000
end

#ThOD

TotalThOD = 0.0;
ThODReg = 75.0;
k = [0.0 0.0 0.0 0.0 0.0 0.0 0.0]



ywasteIsoProp = ([W[1] W[2] W[3]]*transpose([0 0 0.000089]))/Wastea
ywasteHydroA = ([W[1] W[2] W[3]]*transpose([0 0 0.000992]))/Wasteaf
ywastePhenone = ([W[1] W[2] W[3]]*transpose([0 0 0.000068]))/Wastea
ywasteDimethyl = ([W[1] W[2] W[3]]*transpose([0.000049 0 0]))/Waste
ywasteCumene = ([W[1] W[2] W[3]]*transpose([0.000016 0 0.000065]))/
ywastePhenol = ZdischargePhenol;
ywasteAcetone = 0.02 * (W[1]*0 +W[2]*0.013 + W[3]*0.028)/WasteaftTo

TotalThOD = 2.38*10^6*ywastePhenol + 2.2*10^6*ywasteAcetone + 3.2*1
lbofAir = ((TotalThOD - ThODReg)*WasteaftTox*0.4535*29)/(453592*.21
TreatmentCostThOD = lbofAir*0.06*8000
```

```
#pH
WasteRatio = [0.0 0.0 0.0];
pHmean = 0.0;
pOHMean = 0.0;
for i = 1:1:3
    WasteRatio[i] = WARAcetone[i]/TotafterAcRec
end

pH1 = 6.68
pH2 = 6.46
pH3 = 5.69

pHMean = log10(WasteRatio[1]*10^pH1 + WasteRatio[2]*10^pH2 + WasteR
pOHMean = 14 - pHmean

OH = 0.5;
H3O = 0.5;
WH3O = 10^(-pHMean)

WOH = 10^(-pOHMean)
#Water density = 0.4535 l/lb
TotalH3O = WasteaftTox*0.4535*WH3O
TotalOH = WasteaftTox*0.4535*WOH

if pHMean < 7
    IB = 1;
    IA = 0;
    else
    IB = 0;
    IA = 1;
end

#Base Treatment

H3ORemain = (TotalH3O - (IB*TotalOH))
TotalWaste = (WasteaftTox*0.4535)+(IB*TotalOH)
WH3Oafterneutralization = (H3ORemain/TotalWaste)
pHdischarged = -log10(WH3Oafterneutralization)


NaOHCost = 0.31 #$/l
TreatmentBaseCost = IB*TotalOH*0.31*8000;

#Acid Treatment
```

```
OHRemain = (TotalOH - IA*(TotalH3O*H3O*2));
TotalWasteA = (WasteaftTox*0.4535) + (IA*TotalH3O)
WOHafterneutralization = (OHRemain/TotalWasteA)
pOHdischarged = -log10(WOHafterneutralization)

H2SO4Cost = 0.46 #$/l
TreatmentAcidCost = IA*TotalH3O*.46*8000;

TotalTreatmentCost = PhenolToxCost + TreatmentCostThOD + TreatmentA

#display(TotalTreatmentCost)
#display(WasteStreamCalc)

println(TotalTreatmentCost)
    println(WasteStreamCalc)
    r1 = r1 + 1000;
    r2 = r2 + 50;
    r3 = r3 + 500;
    y = y+1;
end
```

```
5941.86279897607
2650.0
10763.277299578152
4200.0
15584.691800180237
5750.0
20406.106300782325
7300.0
25227.520801384395
8850.0
30048.935301986483
10400.0
34870.349802588564
11950.0
39691.76430319064
13500.0
44513.17880379274
15050.0
49334.59330439483
16600.0
```