Unit Commitment, jdm@linux3.ece.iastate.edu

1.0 Introduction

The problem of unit commitment (UC) is to decide which units to interconnect over the next T hours, where T is commonly 24 or 48 hours, although it is reasonable to solve UC for a week at a time. The problem is complicated by the presence of inter-temporal constraints, i.e., what you do in one period constrains what you can do in the next period. The problem is also complicated because it involves integer decision variables, i.e., a unit is either committed (1) or not (0).

The UC problem forms the basis of today's day-ahead markets (DAMs). Most ISOs today are running so-called security-constrained unit commitment (SCUC) 24 hours ahead of the real-time (balancing) market.

If one has a very good solution method to solve the UC problem (or the SCUC problem), then the good solutions that come will save a lot of money relative to using a not-so-good solution method. Regardless of the solution method, however, the solutions may not save much money if the forecast of the demand that needs to be met contains significant error. Having a "perfect" solution for a particular demand forecast is not very valuable if the demand forecast is very wrong. Therefore demand forecasting is very important for solving the UC. Systems that are expecting high wind energy penetrations are concerned about this fact, since high wind penetration increases demand forecast uncertainty (the demand that the thermal units must meet is load-wind). This is why so much attention is being paid to improving wind power forecasting. It is also why so much attention is being paid to creating UC models and solvers that handle uncertainty.

We begin these notes with a motivating example in Section 2.0, then we provide the explicit problem statement, in words, in Section 3.0. Section 4.0 provides the analytic problem statement. Section 5.0 provides an overview of solution methods. Section 6.0 ….
Section 7.0 describes the most important solution method – branch and bound. Section 8.0 illustrates the method on the UC problem. Section 9.0 provides an overview of the SCUC used by several ISOs today.

2.0 Motivating example

Assume we are operating a power system that has load characteristic as given in Fig. 1.

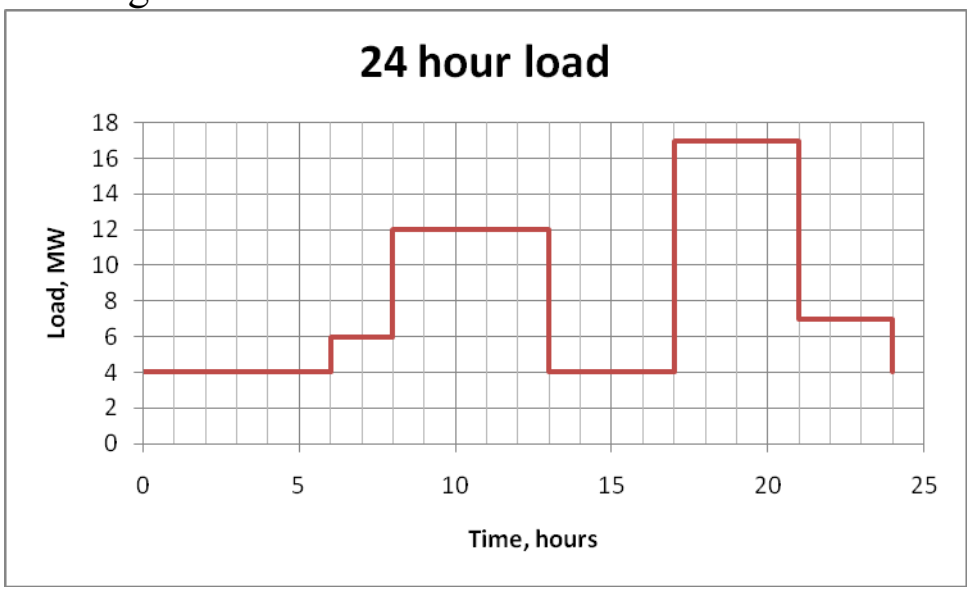


Fig. 1

Consider that we have three units to supply the load. The unit cost rates are expressed below.

$$C_1(P_{g1}) = 5 + 0.5P_{g1} + P_{g1}^2, \quad 0 \le P_{g1} \le 5$$

$$C_2(P_{g2}) = 5 + P_{g2} + 0.5P_{g2}^2, \quad 0 \le P_{g2} \le 10$$

$$C_3(P_{g3}) = 5 + 3P_{g3} + 2P_{g3}^2, \quad 0 \le P_{g3} \le 3$$

Note that the available capacity is 5+10+3=18.

In the economic dispatch problem, we identified the minimum cost for each hour, under the assumption that all units were connected.

The UC problem differs from the economic dispatch problem in that we no longer assume that all of the units are connected. In fact, the essence of the UC problem is to decide which units to connect.

To begin consideration of the problem at hand, let's make the very significant assumption that there are no costs associated with a unit making the transition between up (connected) and down (disconnected).

Therefore our objective is to determine how to operate the three units in order to

- achieve the minimum cost over the 24 hour period and
- satisfy the load.

Let's consider two approaches for doing this.

Approach 1:

In this simple-minded approach, we will connect (commit) all units for the entire 24 hour period, and dispatch them according to economic dispatch at each hour.

Observe that this method will certainly satisfy the load. But it does not achieve minimum cost because, for example, we could simply run unit 1 by itself from 0 to 6 hours and not incur the automatic \$10/hr required by running units 2 and 3 with $P_{g2}=P_{g3}=0$.

So this is a very poor approach.

Approach 2:

Let's try to run only the necessary units for each load-level. But we need to decide which units.

To answer this question, let's consider that there are 7 possible combinations of units. We will denote each combination as S_k. They are enumerated below.

S_1: G_1
S_2: G_2
S_3: G_3
S_4: G_1, G_2
S_5: G_1, G_3
S_6: G_2, G_3
S_7: G_1, G_2, G_3

However, we observe that unit 3 is very expensive therefore let's run this unit only if we must. This means we will eliminate any of the above combinations that have G_3 except the last one. Therefore we now only have four possibilities:

S_1: G_1
S_2: G_2
S_3: G_1, G_2
S_4: G_1, G_2, G_3

We desire to determine which combination should be chosen at each of the various load levels.

To accomplish this, we will plot the total cost of each combination against total load, assuming the units committed are dispatched according to economic dispatch (without losses).

So we want to obtain a function $C_{Tk}(P_d)$ for each set S_k, k=1,2,3,4.

This is easy for S_1 because in this case, $P_d=P_{g1}$, and also for S_2, because in this case, $P_d=P_{g2}$. Therefore, we have

$$C_{T1}(P_d) = 5 + 0.5P_d + P_d^2, \quad 0 \le P_d \le 5$$

$$C_{T2}(P_d) = 5 + P_d + 0.5P_d^2, \quad 0 \le P_d \le 10$$

For S_3 and S_4, we have more than one generator, and so how do we get $C_{T3}(P_d)$?

What we will do here is to write the optimality condition for each generator, which is

$$\lambda = \frac{\partial C_i}{\partial P_{gi}}$$

We will also use

$$P_d = \sum_{i=1}^{N} P_{gi}$$

where N=2 for S_3 and N=3 for S_4.

We do it here for S_3 and just give the result for S_4.

For S_3:

$$C_{T3}(P_d) = 5 + 0.5P_{g1} + P_{g1}^2 + 5 + P_{g2} + 0.5P_{g2}^2$$

$$= 10 + 0.5P_{g1} + P_{g1}^2 + P_{g2} + 0.5P_{g2}^2 \quad , \qquad (\&)$$

Using the optimality condition:

$$, \quad \lambda = \frac{\partial C_1}{\partial P_{g1}} = \frac{\partial C_2}{\partial P_{g2}}$$

we can write that

$$0.5 + 2P_{g1} = 1 + P_{g2} \Rightarrow P_{g2} = 2P_{g1} - 0.5 \qquad (*)$$

From power balance, we have

$$P_d = P_{g1} + P_{g2} \qquad (**)$$

Substitution of (*) into (**) results in

$$P_d = P_{g1} + 2P_{g1} - 0.5 = 3P_{g1} - 0.5$$

Solving for P_{g1} results in

$$P_{g1} = \frac{P_d + 0.5}{3} \qquad (\#)$$

Substitution of (#) into (*) results in

$$P_{g2} = 2P_{g1} - 0.5 = 2\frac{P_d + 0.5}{3} - 0.5 = \frac{4P_d - 1}{6} \quad (\#\#)$$

Substitution of (#) and (##) into (&) above results in

$$C_{T3}(P_d) = 10 + 0.5\left(\frac{P_d + 0.5}{3}\right) + \left(\frac{P_d + 0.5}{3}\right)^2 + \left(\frac{4P_d - 1}{6}\right) + 0.5\left(\frac{4P_d - 1}{6}\right)^2$$

and the above relation is applicable for $0 \le P_d \le 15$.

For S_4:

We will not go through the detailed algebra here but just give the result, which is

$C_{T4}(P_d) =$

$$15 + 0.5\left(\frac{P_d + 1.125}{3.5}\right) + \left(\frac{P_d + 1.125}{3.5}\right)^2 + 0.571P_d + 0.143$$

$$+ 0.5(0.571P_d + 0.143)^2 + 3(0.1429P_d - 0.4643) + 2(0.1429P_d - 0.4643)^2$$

and this relation is applicable for $0 \le P_d \le 18$.

Figure 2 plots C_{T1}, C_{T2}, C_{T3}, and C_{T4} together as a function of demand.

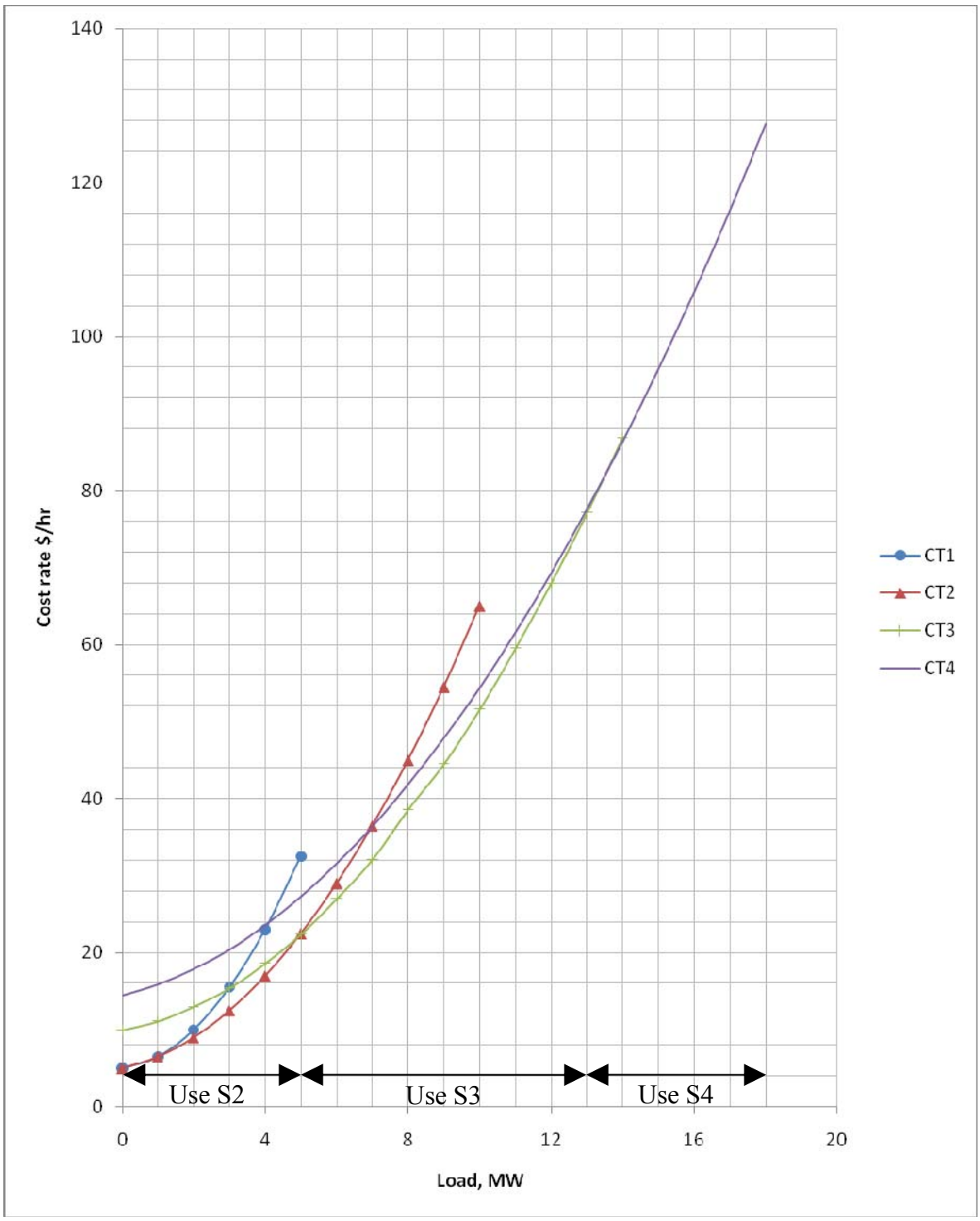


Fig. 2

Recall the generators comprising each set, repeated below for convenience, but now we indicate the load interval for which each set should be used.

S_1: G_1	NEVER
S_2: G_2	0-5
S_3: G_1, G_2	5-13
S_4: G_1, G_2, G_3	13-18

Now we return to the load characteristic of Fig. 1 and use Fig. 2 to identify the "solution" to that particular UC problem. The solution is given in Fig. 3.

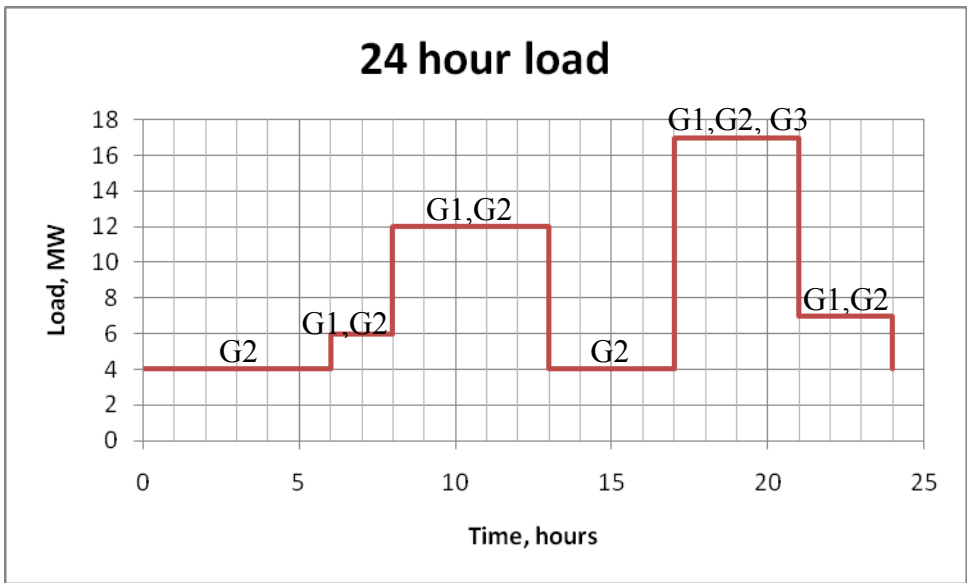


Fig. 3

It is important to note that we have solved this problem under the assumption that transition costs are zero. What if this is not the case?

Transition costs include startup costs and shutdown costs. Startup costs involve both fixed costs and variable costs and requires some further explanation.

Shutdown costs generally involve only fixed costs (mainly labor) and are easy to model. Sometimes they are neglected because they are generally not very significant.

The fixed startup costs (generally labor) will be denoted by C_f. The variable startup costs are denoted by C_V.

Variable costs depends on the shut-down state. There are two possibilities, depending on how "ready" we want the unit to be during its shutdown period. These two possibilities are

- Hot reserve (banking): This is when the unit is down, but the boiler is kept hot. The disadvantage of this state is that it costs money to supply the fuel to heat the boiler. The advantage of this state is the unit can be started quickly. It is also less expensive to start a unit from a hot reserve state since no startup fuel is required to heat the unit. A relation to estimate the variable costs of a hot reserve state is:

$$C_{Vb} = \left[C_b t\right] f$$

  where
  - C_b is the energy per hour to keep the boiler warm (MBTU/hr)
  - f is the cost per MBTU ($/MBTU)
  - t is the shutdown duration

  Note that C_{vb} increases with time, without bound. Therefore the hot reserve state is typically more attractive if the unit will be down for only a short time.
- Cold reserve (cooling): This is when the unit is down and the boiler is not heated. The disadvantage of this state is that the colder the unit is, the more costly and more time to start. The advantage of this state is that there is no fuel cost while the unit is down. A relation to estimate the costs of a cold reserve state is

$$C_{Vc} = C_c \left[1 - e^{-t/\alpha}\right] f$$

  where
  - C_c is the fuel required to start the unit from completely cold (MBTU)
  - α is unit thermal time constant (time constant of thermal loss)
  - f is the cost per MBTU ($/MBTU)

- t is the shutdown duration

Note from the cold reserve equation that

- t=0 implies $C_{Vc}=0$, meaning the unit is not allowed time to cool.
- $t=\infty$ implies $C_{Vc}=C_c f$, meaning the unit becomes completely cold.

3.0 Node-arc model of the UC problem

You will find in the homework problem 1 that when including start-up costs, the peaks allow only one solution (S3) but the valleys allow three which we designate as follows:

S2-H: This is G2 up, with G1 in hot reserve
S2-C: This is G2 up, with G1 in cold reserve
S3: This is G1, G2 up.

You can see that the problem, peaks and valleys, admits only the above three possible states.

We will use a particular representation using nodes and arcs to model the situation where

- node: state of the system at the beginning of a period
- arc: possible path from a state in period i to a state in period i+1.

Notice the use of the term "state" here is to globally specify the status of all units in the system.

For the homework problem, of our three possible states, only one, S_3, is feasible during the peaks when the demand is 11, but all three states are feasible during the valleys when the demand is only 3 or 4. Figures 4 and 5 represents the situation.

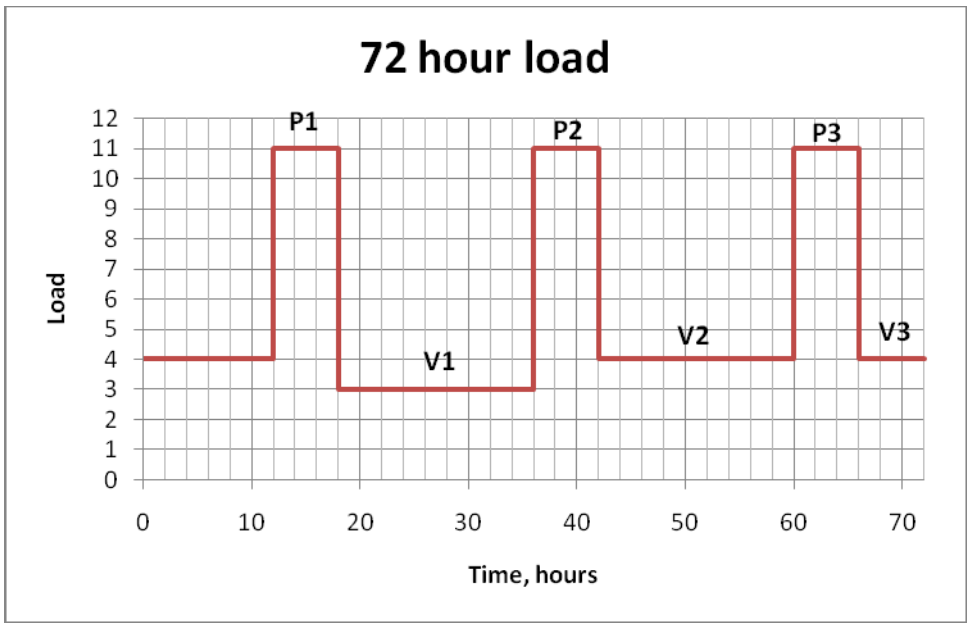

72 hour load
Load
Time, hours
P1
P2
P3
V1
V2
V3
0
1
2
3
4
5
6
7
8
9
10
11
12
0
10
20
30
40
50
60
70

Fig. 4

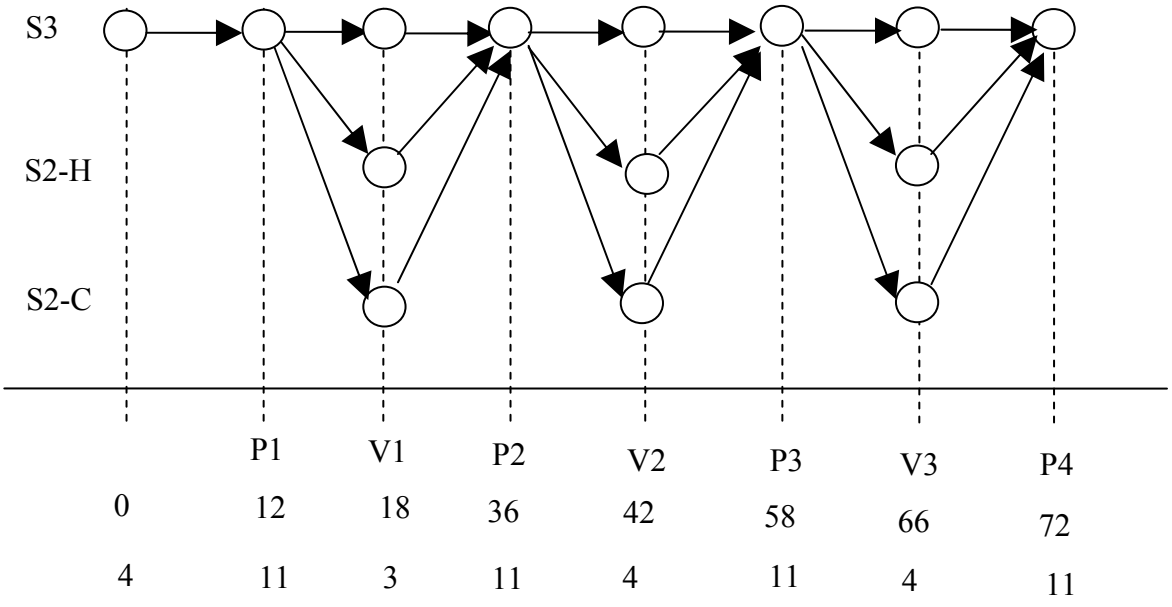

S3
S2-H
S2-C
P1
V1
P2
V2
P3
V3
P4
0
12
18
36
42
58
66
72
4
11
3
11
4
11
4
11

Fig. 5

We can also associate a value with each arc as the cost if the system is in the state from which the arc begins. We can compute these costs for each state and for each different load level.
I will not provide the expressions to make these computations (you will need to do that in your homework). Rather, I will just provide the results in terms of Fig. 6.

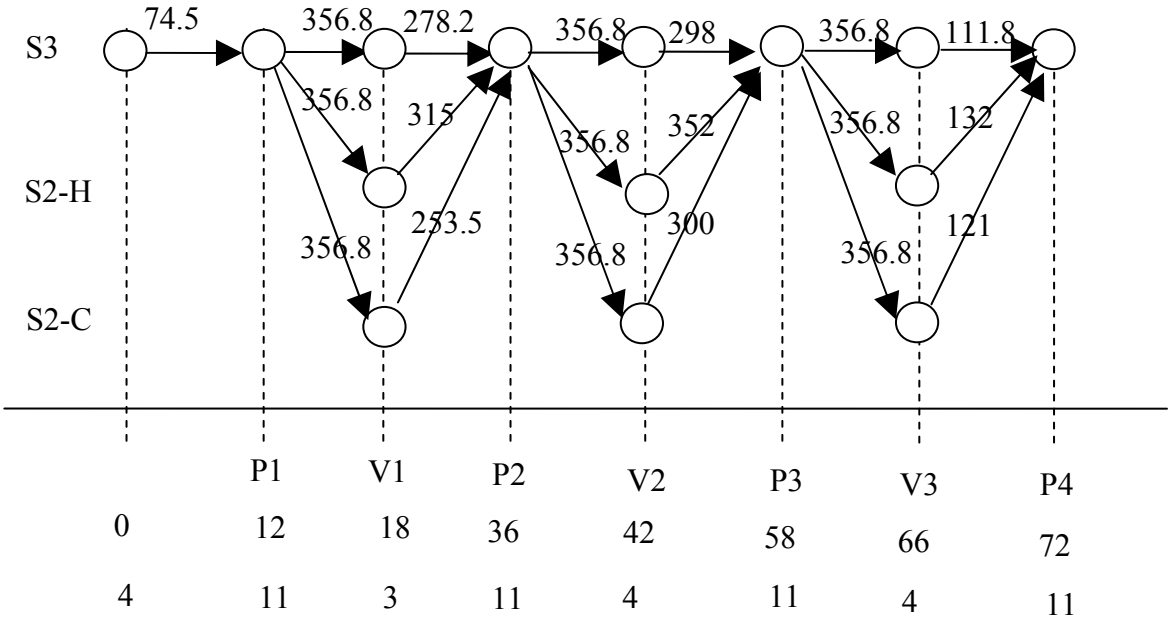


Fig. 6

Observe that the UC problem has been converted to a new problem…

Assume that the values on the arcs are arc-lengths. Then we desire to find the shortest path between the first and last node in the network.

It is easy to see the shortest path in our node-arc model, but consider a case where we have N units instead of 3.

Even if we limit the number of states per unit to two (on or off), then at each loading level, there are 2^N-1 possible states (nodes) to consider, and we still have not thought about the many different transitions (arcs). This represents the **curse of dimensionality**.
If there are m intervals of time for which we must find a solution, each having 2^N-1 possible states, then we will have a total number of possible solutions equal to $(2^N-1)^m$.

For example, consider where N=5 and m=24. In this case $(2^5-1)^{24}$=6.2E35.

Two questions:

1. How do we limit the dimensionality of the problem?
2. How do we algorithmically solve the problem of how to find the shortest path?

Question 1:
There are two approaches:

A. Limit the number of nodes at each time interval.
B. Limit the number of possible transitions (arc) between time intervals.

Example:
Consider 3 gens with 2 possible states (nodes): on or off.
The total number of nodes possible at any time interval is 7.

But let's now prioritize the units using the following rule:

> We always turn on unit i before unit i+1. Therefore, we now only have 3 possible states (nodes), as follows:
>
> $S_1=G_1$
> $S_2=G_1G_2$
> $S_3=G_1G_2G_3$

In general, this rule creates all states $S_i=G_1G_2\ldots G_i$

The prioritization rule is typically done according to economic criteria and security criteria.

Note also that this limits the transitions if you can quantify the maximum possible load variation for one period, see Fig. 7.

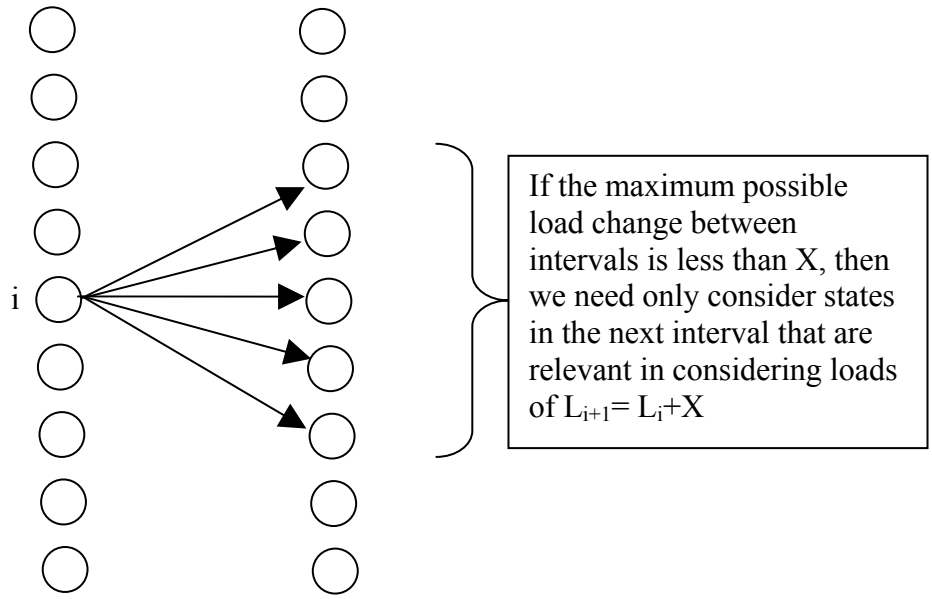


Fig. 7

Question 2: How to algorithmically solve the "shortest path" problem?

Several alternatives:

- Dijkstra's algorithm
- Dial's algorithm
- Label correcting algorithms
- All-pair algorithm
- Dynamic programming: forward and backward

Dynamic programming was used many years ago but not has fallen out to what are called branch and bound methods, which we will explore more fully.

4.0 Problem statement

The unit commitment problem is solved over a particular time period *T*; in the day-ahead market, the time period is usually 24 hours. It is articulated in [10], in words, as follows:

1. Min Objective=UnitEnergyCost+StartupCost+ShutdownCost
   +TransactionCost
   +VirtualBidCost+ DemandBidCost
   +Wheeling Cost

Subject to:

2. Area Constraints:
   a. Demand + Net Interchange
   b. Spinning and Operating Reserves
3. Zonal Constraints:
   a. Spinning and Operating Reserves
4. Security Constraints
5. Unit Constraints:
   a. Minimum and Maximum Generation limits
   b. Reserve limits
   c. Minimum Up/Down times
   d. Hours up/down at start of study
   e. Must run schedules
   f. Pre-scheduled generation schedules
   g. Ramp Rates
   h. Hot, Intermediate, & Cold startup costs
   i. Maximum starts per day and per week
   j. Maximum Energy per day and per study length

We describe the objective function and the various constraints in what follows.

4.1 Objective function

a. *UnitEnergyCost*: This is the total costs of supply over *T*, based on the supply offers made, in $/MWhr.
b. *StartupCost*: This is the total cost of starting units over *T*, based on the startup costs
c. *ShutdownCost*: This is the total cost of shutting down units over T, based on the shutdown costs.
d. *TransactionCost*: Transactions are bilateral agreements made outside the market. Transaction cost for a particular transaction is the difference between nodal prices of transaction sink and source nodes, multiplied by the MW value of the transaction. So TransactionCost is the total transaction costs over *T*.
e. *VirtualBidCost*: Purely financial energy bids and offers made to arbitrage between the day ahead and real time market prices.
f. *DemandBidCost*: This is the total "cost" of demand over *T*, based on the demand bids made, in $/MWhr.
g. *WheelingCost*: I do not find this defined in the PJM materials but assume this is the transmission service cost associated with non-firm transactions.

Revenue from transaction sales, virtual bids and demand bids are added as negative costs so that by minimizing the objective the profit is maximized. For Day Ahead studies, this results in a large negative objective cost.

4.2 Area constraints

a. *Demand + Net Interchange*: The area demand plus the exports from the area (which could be negative, or imports).
b. *Spinning and Operating Reserves*: The spinning reserve is the amount of generation capacity $\Sigma(P_{gmax,k}-P_{gen,k})$ in MW that is on-line and available to produce energy within 10 minutes. Operating reserve is a broader term: the amounts of generating capacity scheduled to be available for specified periods of an Operating Day to ensure the security of the control area. Generally, operating reserve includes primary (which includes spinning) and secondary reserve, as shown in Fig. 8.

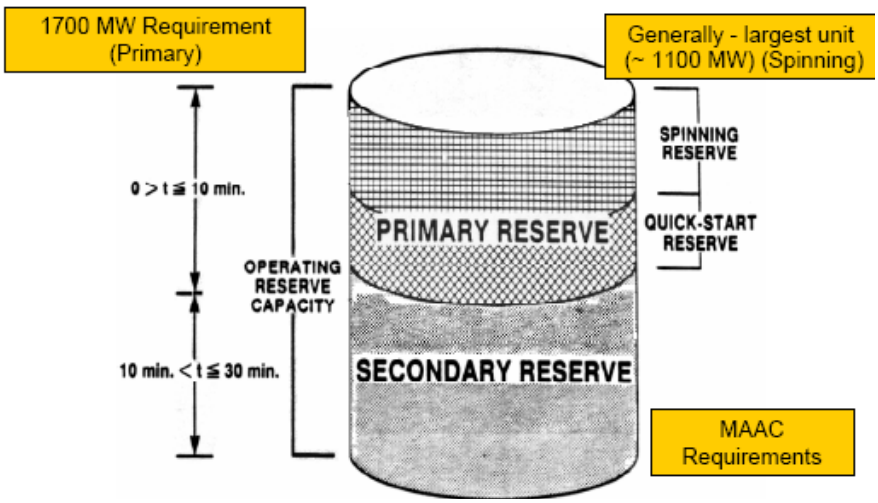


Fig. 8 [1]

4.3 Zonal constraints

Some regions within the control area, called zones, may also have spinning and operating reserve constraints, particularly if transmission interconnecting that region with the rest of the system is constrained.

4.4 Security constraints

These include constraints on branch flows under the no-contingency condition and also constraints on branch flows under a specified set of contingency conditions. The set is normally a subset of all N-1 contingencies.

4.5 Unit constraints

a. *Minimum and Maximum Generation limits*: Self explanatory.
b. *Reserve limits*: The spinning, primary, and/or secondary reserves must exceed some value, or some percentage of the load.
c. *Minimum Up/Down times*: Units that are committed must remain committed for a minimum amount of time. Likewise, units that are de-committed must remain down for a minimum amount of time. These constraints are due to the fact that thermal units can undergo only gradual temperature changes.

d. *Hours up/down at start of study*: The problem must begin at some initial time period, and it will necessarily be the case that all of the units will have been either up or down for some number of hours at that initial time period. These hours need to be accounted for to ensure no unit is switched in violation of its minimum up/down times constraint.
e. *Must run schedules*: There are some units that are required to run at certain times of the day. Such requirements are most often driven by network security issues, e.g., a unit may be required in order to supply the reactive needs of the network to avoid voltage instability in case of a contingency, but other factors can be involved, e.g., steam supply requirements of co-generation plants.
f. *Pre-scheduled generation schedules*: There are some units that are required to generate certain amounts at certain times of the day. The simplest example of this is nuclear plants which are usually required to generate at full load all day. Import, export, and wheel transactions may also be modeled this way.
g. *Ramp Rates*: The rate at which a unit may increase or decrease generation is limited, therefore the generation level in one period is constrained to the generation level of the previous period plus the generation change achievable by the ramp rate over the amount of time in the period.
h. *Hot, Intermediate, & Cold startup costs*: A certain amount of energy must be used to bring a thermal plant on-line, and that amount of energy depends on the existing state of the unit. Possible states are: hot, intermediate, and cold. Although it costs less to start a hot unit, it is more expensive to maintain a unit in the hot state. Likewise, although it costs more to start a cold unit, it is less expensive to maintain a unit in the cold state. Whether a de-committed unit should be maintained in the hot, intermediate, or cold state, depends on the amount of time it will be off-line.
i. *Maximum starts per day and per week*: Starting a unit requires people. Depending on the number of people and the number of units at a plant, the number of times a particular unit may be started in a day, and/or in a week, is usually limited.

j. *Maximum Energy per day and per study length*: The amount of energy produced by a thermal plant over a day, or over a certain study time T, may be less than $P_{max} \times T$, due to limitations of other facilities in the plant besides the electric generator, e.g., the coal processing facilities. The amount of energy produced by a reservoir hydro plant over a time period may be similarly constrained due to the availability of water.

5.0 The UC problem (analytic statement)

The unit commitment problem is a mathematical program characterized by the following basic features.

- *Dynamic*: It obtains decisions for a sequence of time periods.
- *Inter-temporal constraints*: What happens in one time period affects what happens in another time period. So we may not solve each time period independent of solutions in other time periods.
- *Mixed Integer*: Decision variables are of two kinds:
  - Integer variables: For example, we must decide whether a unit will be up (1) or down (0). This is actually a special type of integer variable in that it is binary.
  - Continuous variables: For example, given a unit is up, we must decide what its generation level should be. This variable may be any number between the minimum and maximum generation levels for the unit.

There are many papers that have articulated an analytical statement of the unit commitment problem, more recent ones include [7, 8, 2, 3], but there are also more dated efforts that pose the problem well, although the solution method is not as effective as what we have today, an example is [4].

We provide a mathematical model of the security-constrained unit commitment problem in what follows. This model was adapted from the one given in [5, ch 1]. This model is a mixed integer linear program.

$$\min \underbrace{\sum_t \sum_i z_{it} F_i}_{\text{Fixed Costs}} + \underbrace{\sum_t \sum_i g_{it} C_i}_{\text{Production Costs}} + \underbrace{\sum_t \sum_i y_{it} S_i}_{\text{Startup Costs}} + \underbrace{\sum_t \sum_i x_{it} H_i}_{\text{Shutdown Costs}} \tag{1}$$

subject to

power balance	$\sum_i g_{it} = D_t = \sum_i d_{it}$	$\forall\, t,$	(2)
reserve	$\sum_i r_{it} = SD_t$	$\forall\, t,$	(3)
min generation	$g_{it} \geq z_{it} MIN_i$	$\forall\, i,t,$	(4)
max generation	$g_{it} + r_{it} \leq z_{it} MAX_i$	$\forall\, i,t,$	(5)
max spinning reserve	$r_{it} \leq z_{it} MAXSP_i$	$\forall\, i,t,$	(6)
ramp rate pos limit	$g_{it} \leq g_{it-1} + MxInc_i$	$\forall\, i,t,$	(7)
ramp rate neg limit	$g_{it} \geq g_{it-1} - MxDec_i$	$\forall\, i,t,$	(8)
start if off-then-on	$z_{it} \leq z_{it-1} + y_{it}$	$\forall\, i,t,$	(9)
shut if on-then-off	$z_{it} \geq z_{it-1} - x_{it}$	$\forall\, i,t,$	(10)
normal line flow limit	$\sum_i a_{ki}(g_{it} - d_{it}) \leq MxFlow_k$	$\forall\, k,t,$	(11)
security line flow limits	$\sum_i a_{ki}^{(j)}(g_{it} - d_{it}) \leq MxFlow_k^{(j)}$	$\forall\, k,j,t,$	(12)

where the decision variables are:

- g_{it} is the MW produced by generator i in period t,
- r_{it} is the MW of spinning reserves from generator i in period t,
- z_{it} is 1 if generator i is dispatched during t, 0 otherwise,
- y_{it} is 1 if generator i starts at beginning of period t, 0 otherwise,
- x_{it} is 1 if generator i shuts at beginning of period t, 0 otherwise,

Other parameters are

- D_t is the total demand in period t,
- SD_t is the spinning reserve required in period t,
- F_{it} is fixed cost ($/period) of operating generator i in period t,
- C_{it} is prod. cost ($/MW/period) of operating gen i in period t;
- S_{it} is startup cost ($) of starting gen i in period t.
- H_{it} is shutdown cost ($) of shuting gen i in period t.
- $MxInc_i$ is max ramprate (MW/period) for increasing gen i output
- $MxDec_i$ is max ramprate (MW/period) for decreasing gen i output

- a_{ij} is linearized coefficient relating bus i injection to line k flow
- $MxFlow_k$ is the maximum MW flow on line k
- $a_{ki}^{(j)}$ is linearized coefficient relating bus i injection to line k flow under contingency j,
- $MxFlow_k^{(j)}$ is the maximum MW flow on line k under contingency j

The above problem statement is identical to the one given in [5] with the exception that here, we have added eqs. (11) and (12).
➔The addition of eq. (11) alone provides that this problem is a transmission-constrained unit commitment problem.
➔ The addition of eqs. (11) and (12) together provides that this problem is a security-constrained unit commitment problem.

One should note that our problem is entirely linear in the decision variables. Therefore this problem is a *linear* mixed integer program, and it can be compactly written as

$$\min \underline{c}^T \underline{x}$$

Subject to

$$\underline{A}\underline{x} \leq \underline{b}$$

We have already had some discussion about solution methods to the above problem, in that we have illustrated priority list methods and we have described dynamic programming and Lagrangian Relaxation. There is one more to discuss. In summary, then, there have four basic solution methods used in the past few years:

- Priority list methods
- Dynamic programming
- Lagrangian relaxation
- Branch and bound

The last method, branch and bound, is what the industry means when it says "MIP." It is useful to understand that the chosen method can have very large financial implications. This point is well-made in the chart [6] of Fig. 9.

	Current Approach	Planned Approach	Date of Planned Implementation of MIP	Estimated Annual Savings
Real-time market look ahead	LR used for 2 hour look ahead commitment and dispatch	MIP: 2 hour look ahead for dispatch. As long as 5 hours for commitment .	April 1, 2008	~$100,000-$1 million (0.1%-1%1 of 2006 RT Dispatch Costs and RT RMR Costs2: $97 million)
Residual unit commitment	Procedural based operator judgement advised by a MIP based UC with no network	Run a MIP, Full Network Model based on Residual Unit Commitment after Day-Ahead bid market.	April 1, 2008	~$100,000-$1 million (based on 0.1% - 1% of Total Minimum Load Costs for 2006: $106 million)
day-ahead market	Linear Programing: No unit commitment, No Energy Optimziation, Allocation of Transmission only using zonal model	Run a MIP based SCUC/SCED, Full Network Model program, Energy and A/S co-optimized	April 1, 2008	~$2.3-$23 million (Assumes an estimated 0.1%-1% reduction of $11.4 billion Energy and Ancillary Service)
Capacity market	None	Policy being considered	Policy being considered	No Estimate
Ancillary service market	Linear Programming sequential procured after Transmission Allocation	Run a MIP based SCUC/SCED, Full Network Model program co-optimized with energy	April 1, 2008	~$230,000-$2.3 million (0.1%-1%[1] of 2006 A/S costs[2] of $234 million)
planning	Powerflow studies	No immediate plans to incorporate MIP	No immediate plans to incorporate MIP	No Estimate

Fig. 9

6.0 UC and Day-ahead market

The main tool used to implement the day-ahead-markets (DAM) is the security-constrained unit commitment program, or SCUC. In this section, we review some basics about the DAM by looking at some descriptions given by a few industry authors. You are encouraged to review the papers from which these quotes were taken. Notice that any references made inside the quotations are given only in the bibliography of the subject paper and not in the bibliography of these notes. References made outside of the quotations are given in the bibliography of these notes.

6.1 Paper by Chow & De Mello:

Reference [7] offers an overall view of the sequence of functions used by an ISO, as given in Fig. 10. Observe that the "day-ahead

scheduling" and the "real time commitment and dispatch" both utilize the SCUC.

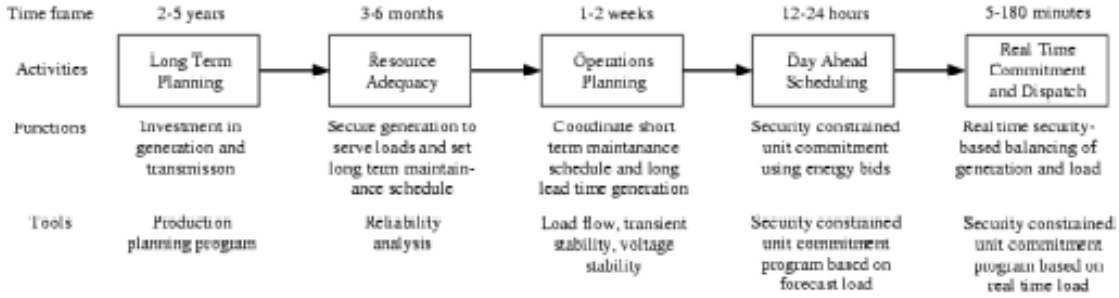


Fig. 10

They state:
"Electricity is a commodity that cannot be effectively stored and the energy-supplying generators have limits on how quickly they can be started and ramped up or down. As a result, both the supply and demand become more inelastic and the electricity market becomes more volatile and vulnerable as it gets closer to real time [34]. To achieve a stable margin as well as to maintain the system reliability, a forward market is needed to provide buyers and sellers the opportunity to lock in energy prices and quantities and the ISO to secure adequate resources to meet predicted energy demand well in advance of real time. Thus architecturally, many ISOs (e.g. PJM, ISO New England, New York ISO) take a multisettlement approach for market design…."

"The two main energy markets, each producing a financial settlement, in a multisettlement system, are the following.
1) DAM: schedules resources and determines the LMPs for the 24 h of the following day based on offers to sell and bids to purchase energy from the market participants.
2) Real-time market: optimizes the clearing of bids for energy so that the real-time system load matching and reliability requirements are satisfied based on actual system operations. LMPs are computed for settlement at shorter intervals, such as 5–10 min…."

"Fig. 6 shows the timeline of the multiple-settlement systems used in NYISO, PJM, and ISO-NE, which are typical of those used in practice. Supply and demand bids are submitted for the DAM, typically 12–24 h ahead of the real-time operation. Then the day-ahead energy prices are computed and posted, 6–12 h ahead of real-time operation…."

"The DAM typically consists of supply and demand bids on an hourly basis, usually from midnight to the following midnight. The supply bids include generation supply offers with start-up and no-load costs, incremental and decremental bids[1], and external transactions schedules. The demand bids are submitted by loads individually or collectively through load-serving entities. In scheduling the supply to meet the demand, all the operating constraints such as transmission network constraints, reserve requirements, and external transmission limits must not be violated. This process is commonly referred to as an SCUC problem, which is to determine hourly commitment schedules with the objective of minimizing the total cost of energy, start-up, and spinning at no-load while observing transmission constraints and physical resources' minimum runtime, minimum downtime, equipment ramp rates, and energy limits of energy-constrained resources. Based on the commitment schedules for physical resources, SCUC is used to clear energy supply offers, demand bids, and transaction schedules, and to determine LMPs and their components at all defined price nodes including the hubs, zones, and aggregated price nodes for the DAM settlement. The SCUC problem is usually optimized using a Lagrangian relaxation (LR) or a mixed-integer programming (MIP) solver…."

"A critical part of the DAM is the bid-in loads, which is a day-ahead forecast of the real-time load. The load estimate depends on the

[1] Decremental bids are similar to price-sensitive demand bids. They allow a marketer or other similar entity without physical demand to place a bid to purchase a certain quantity of energy at a certain location if the day-ahead price is at or below a certain price. Incremental offers are the flip side of decremental bids.

season, day type (weekday, weekend, holiday), and hour of the day. Most ISOs have sophisticated load forecasting programs, some with neural network components [36], [37], to predict the day-ahead load to within 3%–5% accuracy and the load forecasts are posted. LSEs with fully hedged loads through long-term bilateral contracts tend to bid in the amount corresponding to the ISO predicted loads. Some other LSEs may bid in loads that are different from those posted by the ISO. In such cases, if the LSE bid load exceeds the ISO load, the LSE bid load is taken as the load to be dispatched. Otherwise, the ISO load will supersede the LSE bid load and the SCUC will commit generators to supply the ISO forecasted load in a reliability stage. Then the generation levels of the committed generators will be allocated to supply LSE bid loads. Committing extra generators outside the DAM will be treated as uplifts and be paid by the LSEs…."

6.2 Paper by Papalexopoulos:

Reference [8] states:

"The Must Offer Waiver (MOW) process is basically a process of determining which Must Offer units should be committed in order to have enough additional capacity to meet the system energy net short which is the difference between the forecast system load and the Day-Ahead Market energy schedules. This commitment process ensures that the resulting unit schedule is feasible with respect to network and system resource constraints. Mathematically, this can be stated as a type of a SCUC problem [3]. The objective is to minimize the total start up and minimum load costs of the committed units while satisfying the power balance constraint, the transmission interface constraints, and the system resource constraints, including unit inter-temporal constraints…."

"The most popular algorithms for the solutions of the unit commitment problems are Priority-List schemes [4], Dynamic Programming [5], and Mixed Integer Linear Programming [6]. Among these approaches the MILP technique has achieved

significant progress in the recent years [7]. The MILP methodology has been applied to the SCUC formulation to solve this MOW problem. Recent developments in the implementation of MILP-based algorithms and careful attention to the specific problem formulation have made it possible to meet accuracy and performance requirements for solving such large scale problems in a practical competitive energy market environment. In this section the MILP-based SCUC formulation is presented in detail…."

6.3 Paper by Ott:

Reference [9] states:
"In addition to the LMP concept, the fundamental design objectives of the PJM day-ahead energy market are: 1) to provide a mechanism in which all participants have the opportunity to lock in day-ahead financial schedules for energy and transmission; 2) to coordinate the day-ahead financial schedules with system reliability requirements; 3) to provide incentive for resources and demand to submit day-ahead schedules; and 4) to provide incentive for resources to follow real-time dispatch instructions…."

6.4 Paper by AREVA and PJM:

Reference [10] states:
"As the operator of the world's largest wholesale market for electricity, PJM must ensure that market-priced electricity flows reliably, securely and cost-effectively from more than 1100 Generating resources to serve a peak load in excess of 100,000 MW. In doing so, PJM must balance the market's needs with thousands of reliability-based constraints and conditions before it can schedule and commit units to generate power the next day. The PJM market design is based on the Two Settlement concept [4]. The Two-Settlement System provides a Day-ahead forward market and a real-time balancing market for use by PJM market participants to schedule energy purchases, energy sales and bilateral contracts. Unit

commitment software is used to perform optimal resource scheduling in both the Day-ahead market and in the subsequent Reliability Analysis…."

"As the market was projected to more than double its original size, PJM identified the need to develop a more robust approach for solving the unit commitment problem. The LR algorithm was adequate for the original market size, but as the market size increased, PJM desired an approach that had more flexibility in modeling transmission constraints. In addition, PJM has seen an increasing need to model Combined-cycle plant operation more accurately. While these enhancements present a challenge to the LR formulation, the use of a MIP formulation provides much more flexibility. For these reasons, PJM began discussion with its software vendors, in late 2002, concerning the need to develop a production grade MIP-based approach for large-scale unit commitment problems…."

"The Day-ahead market clearing problem includes next-day generation offers, demand bids, virtual bids and offers, and bilateral transactions schedules. The objective of the problem is to minimize costs subject to system constraints. The Day-ahead market is a financial market that provides participants an operating plan with known compensation: If their generation (or load) is the same in the real-time market, their revenue (or cost) is the same. Compensation for any real-time deviations is based on real-time prices, providing participants with opportunities to improve profit (or reduce cost) if they have flexibility to adjust their schedules…."

"In both problems, unit commitment accepts data that define bids (e.g., generator constraints, generator costs, and costs for other resources) and the physical system (e.g., load forecast, reserve requirements, security constraints). In real time, the limited responsiveness of units and additional physical data (e.g., state

estimator solution, net-interchange forecast) further constrains the unit commitment problem."

"The Unit Commitment problem is a large-scale non-linear mixed integer programming problem. Integer variables are required for modeling: 1) Generator hourly On/Off-line status, 2) generator Startups/Shutdowns, 3) conditional startup costs (hot, intermediate & cold). Due to the large number of integer variables in this problem, it has long been viewed as an intractable optimization problem. Most existing solution methods make use of simplifying assumptions to reduce the dimensionality of the problem and the number of combinations that need to be evaluated. Examples include priority-based methods, decomposition schemes (LR) and stochastic (genetic) methods. While many of these schemes have worked well in the past, there is an increasing need to solve larger (RTO-size) problems with more complex (e.g. security) constraints, to a greater degree of accuracy. Over the last several years, the number of units being scheduled by RTOs has increased dramatically. PJM started with about 500 units a few years ago, and is now clearing over 1100 each day. MISO cases will be larger still…."

"The classical MIP implementation utilizes a Branch and Bound scheme. This method attempts to perform an implicit enumeration of all combinations of integer variables to locate the optimal solution. In theory, the MIP is the only method that can make this claim. It can, in fact, solve non-convex problems with multiple local minima. Since the MIP methods utilize multiple Linear Programming (LP) executions, they have benefited from recent advances in both computer hardware and software [6]…"

"This section presents results from using the CPLEX 7.1 and CPLEX 9.0 MIP solvers on a large-scale RTO Day Ahead Unit Commitment problem. This problem has 593 units and a 48 hour time horizon…."

7.0 Solution methods for mixed integer programs

To begin with, we need to discuss a very fundamental principle related to linear programming. Consider a standard linear program, given below. Notice that variables are continuous, not integer.

$$\max Z = x_1 + 5x_2$$

$$s.t.$$

$$x_1 + 10x_2 \le 20$$

$$x_1 \le 2$$

$$x_1 \ge 0, x_2 \ge 0$$

We can use Matlab-linprog or CPLEX to solve this, but it is so simple that you we can also visualize the solution in x_1-x_2 space, as in Fig. 11 below. The shaded area indicates the feasible region.

Observe the following in Fig. 11:

- The parallel lines correspond to different values of the objective function.
- The objective function is increasing from bottom to top, as indicated by the heavy arrow. The parallel lines are contours of constant Z. For example, note there are two dark dotted parallel lines. The lower one corresponds to Z=5. The upper one corresponds to Z=10.
- The solid dark parallel line is the highest Z-contour (parallel line) which touches one feasible point.
- Unless the Z-contours are parallel to a binding constraint, the highest Z-contour which touches only one feasible point will always do so at a *corner point*. A corner point is a point of intersection between two or more constraints.

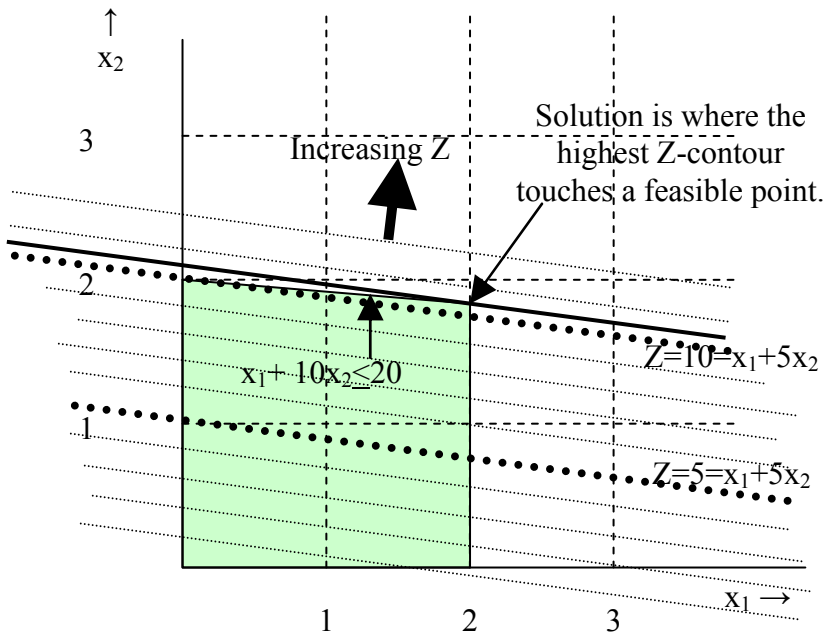


Fig. 11

7.1 Integer programs

In Sections 5 and 6, we observed that our problem is a linear mixed integer programming problem. Before we discuss our chosen method of solving this problem, it is useful to see where our problem lies in the general area of optimization.

Linear mixed integer programs are a particular type of problem that falls under the more general heading of integer programs (IPs). IPs may be *pure*, in which case all the decision variables are integer (PIP), or they may be *mixed*, in which case some decision variables are integer and some are continuous valued (MIP).

It is also possible to have problems where the integer variables may take on one of just two values: 0 and 1. Such problems occur frequently when the decisions to be made are of the "yes or no" type. When all integer variables are this way, the problem is considered to be a binary integer program (BIP).

The remaining material in this section is adapted from [11, ch 13.3].

There are two obvious approaches that come to mind for solving IPs. One is to check every possible solution. We call this exhaustive enumeration. Another is to solve the problem as a linear program without integrality requirements, and then round the values we get to the nearest integer. We call this LP-relaxation with rounding. Let's look at these two approaches.

7.1.1 Exhaustive enumeration

Consider a BIP problem with 3 variables: x_1, x_2, and x_3, each of which can be 1 or 0. The possible solutions are

(0,0,0), (0,0,1), (0,1,0), (0,1,1),
(1,0,0), (1,0,1), (1,1,0), (1,1,1).

Thus, there are $2^3=8$ solutions. It would not be too hard to check them all. But consider more typical problems with 30 variables, or even 300.

$2^{30}=1.0737\times10^9=1{,}073{,}700{,}000$ (over a billion possible solutions)
$2^{300}=2.037\times10^{90}$

Recall our UC formulation in the previous notes. There were three integer variables for every generator:

- z_{it} is 1 if generator i is dispatched during t, 0 otherwise,
- y_{it} is 1 if generator i starts at beginning of period t, 0 otherwise,
- x_{it} is 1 if generator i shuts at beginning of period t, 0 otherwise,

PJM, for example, has over 1100 generators. For PJM's unit commitment problem, the number of integer decision variables is over 3300. We see there are over 2^{3300} possible solutions! It is for this reason, as we have seen in Section 3, that the UC problem is said to suffer from the "curse of dimensionality."

The conclusion is that exhaustive enumeration is simply not feasible, even for the fastest computers.

7.1.2 LP relaxation with rounding

Here, we relax the requirement that the decision variables be integers. Then our problem becomes a standard LP, and we can solve it efficiently using the simplex method. When we examine the resulting solution, some or all of the variables are likely to be non-integer. These variables are then just rounded to the nearest feasible integer, and we call what results our solution.

This approach may in fact work with reasonable accuracy (it may get close to the optimum) for some problems, especially if the values of the variables are large so that rounding creates relatively little error. (This would not be the case, however, for BIPs, as in the case of the UC, where integer variables are either 1 or 0.)

But there are two pitfalls in this approach.

Pitfall #1: The solution obtained by rounding the optimal LP solution is not necessarily feasible. To illustrate, consider a problem having the below two constraints.

$$-x_1 + x_2 \leq 3.5$$

$$x_1 + x_2 \leq 16.5$$

Assume the LP-relaxation found a solution that has x_1=6.5 and x_2=10, so that

$$-6.5 + 10 = 3.5 \leq 3.5$$

$$6.5 + 10 = 16.5 \leq 16.5$$

If we round x_1 to 7, then

$$-7 + 10 = 3 \leq 3.5$$

$$7 + 10 = 17 \text{ not} \leq 16.5$$

If we round x_1 to 6, then

$$-6 + 10 = 4 \text{ not} \leq 3.5$$

$$6 + 10 = 16 \leq 16.5$$

Thus, either way we go, rounding up or down, we result in an infeasible solution. The only way we can make x_1 an integer is if we also change x_2. This situation is illustrated using Fig. 12 below.

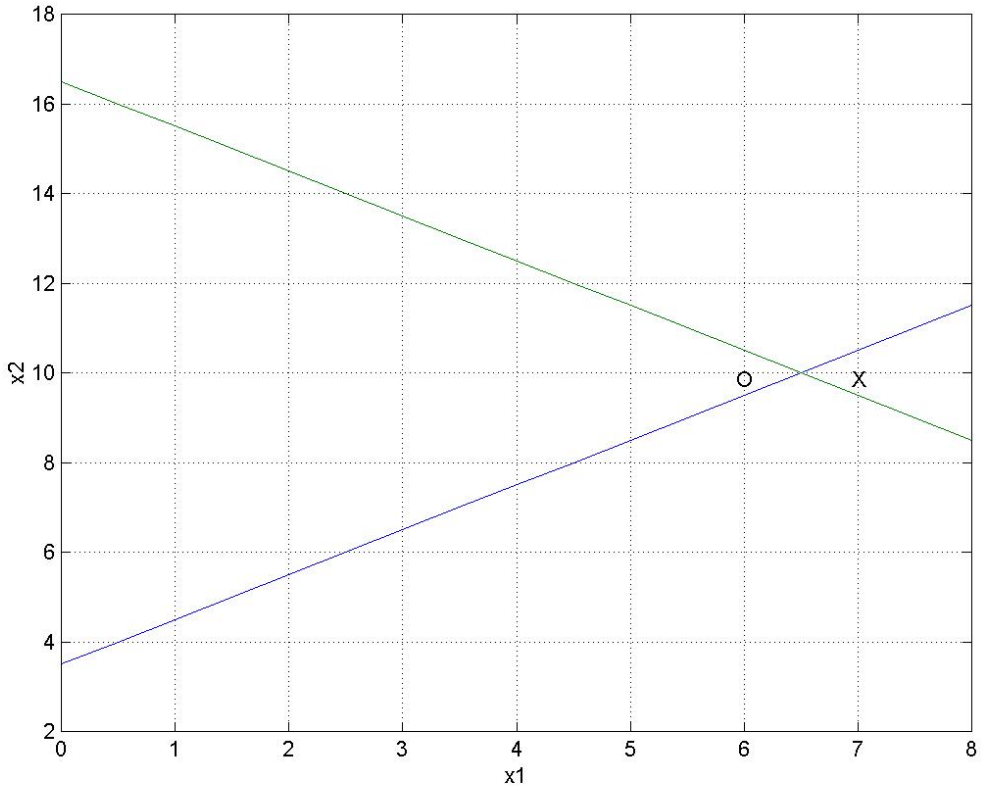


Fig. 12

In Fig. 12, the feasible region is between the x-axis and the two constraint-lines. The "corner point" of (6.5, 10) is the solution to the relaxed IP. When x1 is rounded to 7, the solution is X in Fig. 12, which is clearly above the feasible region. When x1 is rounded to 6, the solution is O in Fig. 12, again, clearly above the feasible region.

Pitfall #2: Even if the rounded solution is feasible, there is no guarantee it will be optimal or that it will even be reasonably accurate (reasonably close to the optimal). This is illustrated by the following problem.

$$\max Z = x_1 + 5x_2$$

$$s.t.$$

$$x_1 + 10x_2 \le 20$$

$$x_1 \le 2$$

$$x_1 \ge 0, x_2 \ge 0$$

$$x_1, x_2 \text{ integers}$$

This problem is illustrated in Fig. 12.

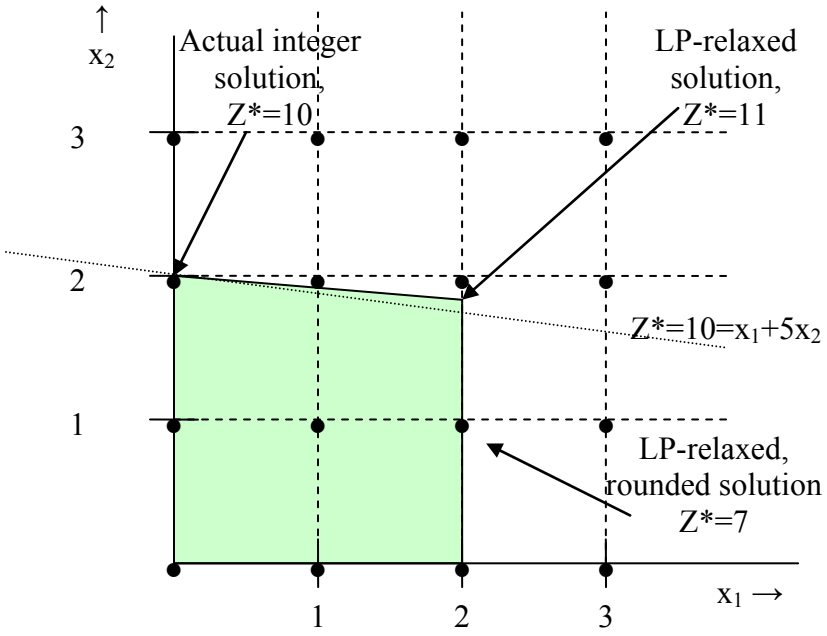


Fig. 12

In Fig. 12, the dots are the possible integer solutions, and the shaded region is the feasible region. The LP-relaxed solution is (2, 9/5) where Z*=11. If we rounded, then we would get (2, 1) where Z*=7. But it is easy to see that the point (0, 2) is on the Z*=10 line. Because (2,0) is integer and feasible, and higher than Z*=7, we see that the rounding approach has failed us miserably.

7.2 Some other methods

There are two broad classes of methods to solving IPs:

- Cutting plane methods
- Tree-search methods

7.2.1 Cutting plane methods

Cutting plane methods generate additional constraints. There are several kinds of cutting plane methods. One of the simplest to understand is the fractional integer programming algorithm [12, ch 13]. In this method, we shrink the feasible region the minimum amount possible so that all corner points are integer. If we are successful in doing this, then the solution to the corresponding relaxed LP will be integer and thus the solution to the original integer program. Figure 13 illustrates.

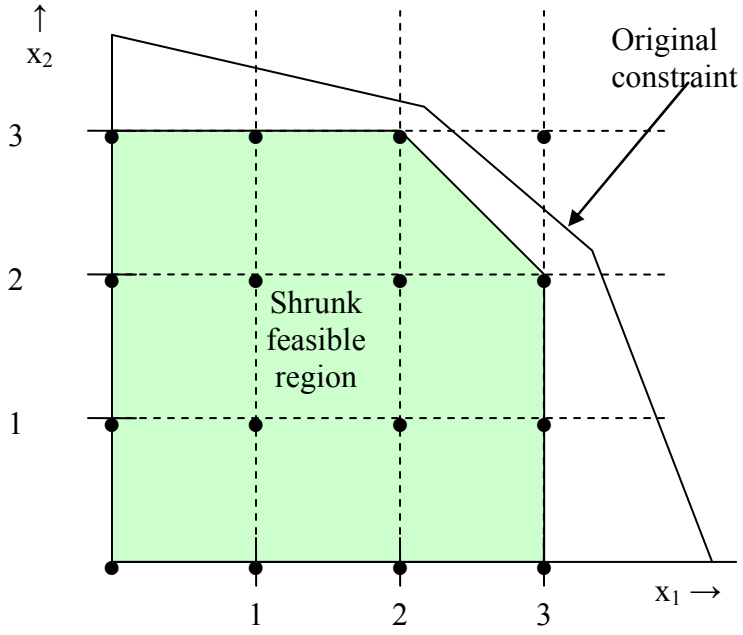


Fig. 13

Another very popular cutting plane method, particularly for MIP, is Benders decomposition [12, Sec 15.2]. To apply this method, the variables must be separable. If they are, one can set up a Master problem to solve for one set of variables and subproblems to solve for the other set. Then the algorithm iterates between Master and subproblems.

7.2.2 Tree-search methods

The essence of tree-search algorithms are that they conceptualize the problem as a huge tree of solutions, and then they try to do some smart things to avoid searching the entire tree. Such a tree is shown in Fig. 14.

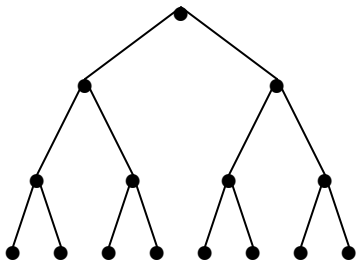


Fig. 14

The common features of tree-search algorithms are [12, App C]:

1. They are easy to understand;
2. They are easy to program on a computer;
3. The upper bound on the number of steps the algorithm needs to find the solution is $O(k^n)$, where n is the number of decision variables (this means running time increases exponentially with the number of variables);
4. They lack mathematical structure.

The most popular tree-search method today is called branch and bound. We will study this method in the next section.

7.3 Branch and bound (B&B)

Two definitions are necessary.

- Predecessor: Pj
- Successor: Pk

Problem Pj is predecessor to Problem Pk, and Problem Pk is successor to Problem Pj, if they are identical with the exception that

one continuous-valued variable in Problem Pj is constrained to be integer in Problem Pk.

How to constrain a continuous-valued variable to be integer?
Consider the following problem [13, ch. 23]. We will call it P_0.

$$\max \quad \zeta = 17x_1 + 12x_2$$

s.t.

$$P_0 \qquad 10x_1 + 7x_2 \le 40$$

$$x_1 + x_2 \le 5$$

$$x_1, x_2 \ge 0$$

Solution using CPLEX yields

$$P_0 \quad Solution: \qquad x_1 = 1.667, x_2 = 3.333, \qquad \zeta = 68.333$$

Observe this solution as the corner point labeled "solution" in Fig. 15.

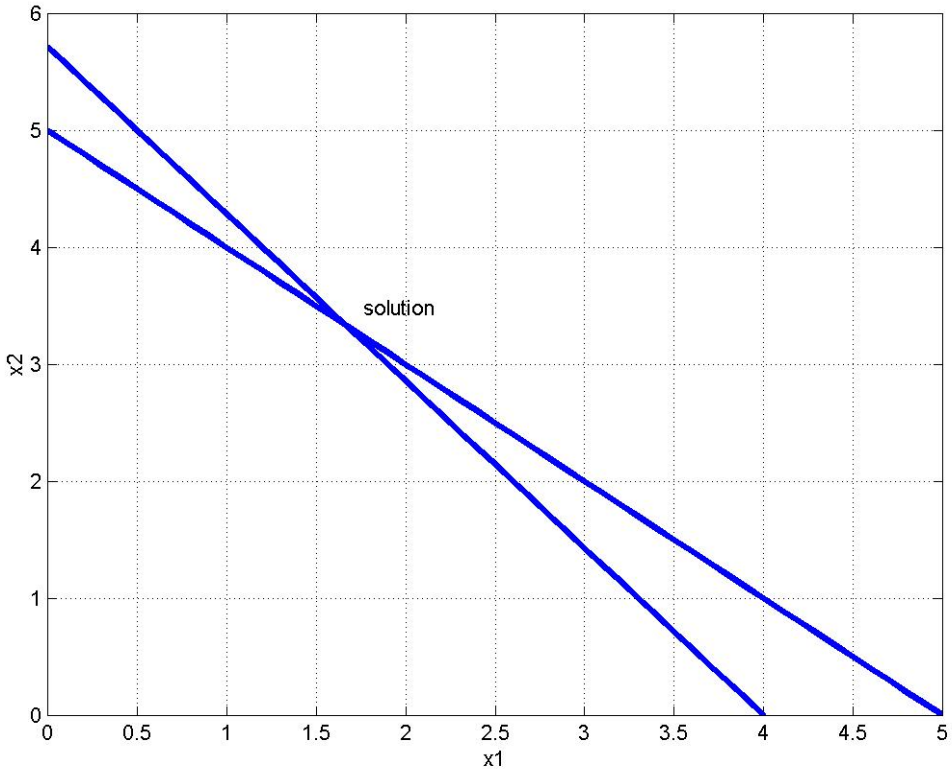


Fig. 15

What if we now pose a problem P_1 to be exactly like P_0 except that we will constrain $x_1 \leq 1$? Here it is:

$$\begin{aligned} &\max \quad \zeta = 17x_1 + 12x_2 \\ &\text{s.t.} \\ P_1 \qquad &10x_1 + 7x_2 \leq 40 \\ &x_1 + x_2 \leq 5 \\ &x_1 \leq 1 \\ &x_1, x_2 \geq 0 \end{aligned}$$

What do you expect the value of x_1 to be in the optimal solution?

Because the solution without the constraint $x_1 \leq 1$ wanted 1.667 of x_1, we can be sure that the solution with the constraint $x_1 \leq 1$ will want as much of x_1 as it can get, i.e., it will want $x_1 = 1$.

But now let's ask, in terms of constraining a continuous-valued variable to be integer: what have we done here?

1. We solved a *predecessor* problem as a relaxed LP and obtained optimal values (but non-integer) for the variables.
2. Then we constructed a *successor* problem by indirectly imposing integrality on one variable via constraining it to be less than or equal to the integer *just* less than the value of that variable in the optimal solution to the predecessor problem.

Let's use CPLEX to solve P_1 to see if it works. The CPLEX solution is:

$$P_1 \quad \textit{Solution}: \qquad x_1 = 1.0, x_2 = 4.0, \qquad \zeta = 65.0$$

It worked in that x_1 did in fact become integer. In fact, x_2 became integer as well, but this is by coincidence, i.e., in general, the "trick" of imposing integrality on a variable, as we have done above, is not guaranteed to also impose integrality on the remaining variables.

A related way to think of this is that we just made a new corner point, as illustrated in Fig. 16, that had to be the next "best" (largest

ζ in this case) corner point to the optimal without the constraint $x_1 \leq 1$.

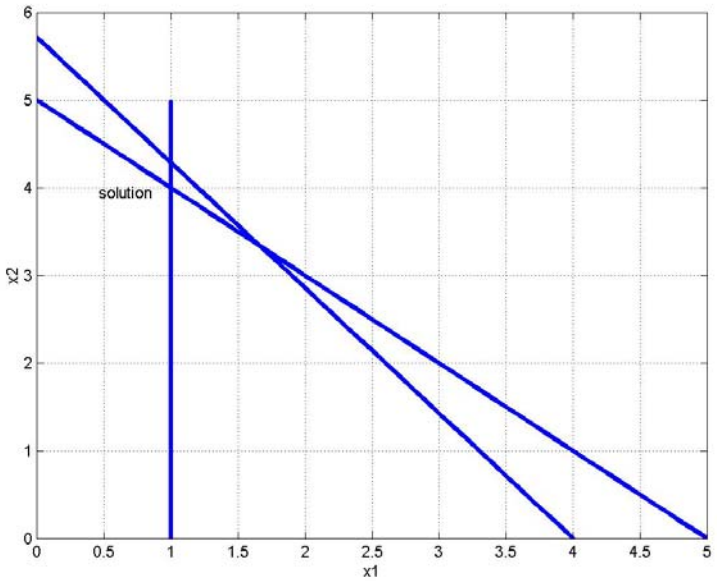


Fig. 16

So we understand now how to constrain a continuous-valued variable to be integer.

But here is another question for you… What if we want to solve the following IP:

$$\max \quad \zeta = 17x_1 + 12x_2$$

s.t.

$$IP_1 \qquad \begin{aligned} 10x_1 + 7x_2 &\leq 40 \\ x_1 + x_2 &\leq 5 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\text{ integers.} \end{aligned}$$

This problem is identical to P_0 except we require x_1, x_2 to be integer. The question is: Is the P_1 solution we obtained, which by chance is a *feasible* solution to IP_1, also the *optimal* solution to IP_1?

To answer this question, let's state a rather obvious criterion.

> ***IP Optimality Criterion***: A solution to an IP is optimal if the corresponding objective value is better than the objective value corresponding to every other feasible solution to that IP.

So is the P_1 solution the optimal solution to IP_1?

Answer: We do not know. Why?

➔ Because we have not yet explored the entire solution space.

What part of the solution space remains?

Answer: The part associated with $x_1=2$. So let's constrain $x_1 \geq 2$. This results in problem P_2.

$$
\begin{aligned}
& \max \quad \zeta = 17x_1 + 12x_2 \\
& \text{s.t.} \\
P_2 \qquad & 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 \\
& x_1 \geq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Using CPLEX to solve P_2 results in:

$$P_2 \quad Solution: \quad x_1 = 2.0, x_2 = 2.857, \qquad \zeta = 68.8286$$

This time, we were not so fortunate to obtain a feasible solution to IP_1, since x_2 is not integer. And so the P_2 solution is not feasible to IP_1, and therefore it is certainly not optimal to IP_1. The situation is illustrated in Fig. 17.

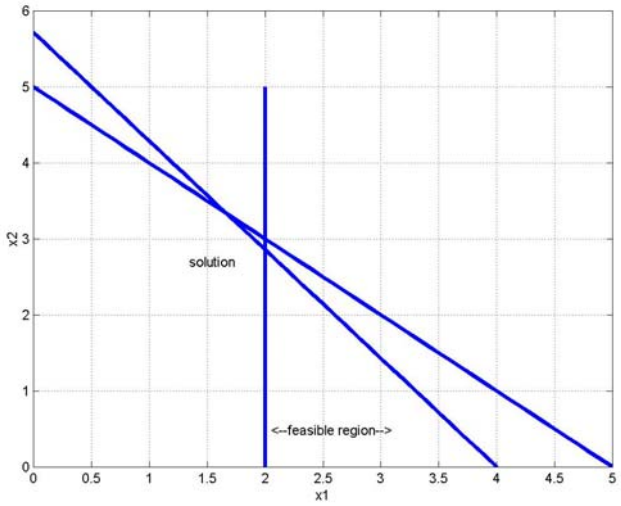


Fig. 17

But does the P_2 solution tell us anything useful?

YES! Compare the objective function value of P_2, which is 68.8286, with the objective function value of P_1, which is 65. Since we are maximizing, the objective function value of P_2 is better. But the P_2 solution is not feasible. However, we can constrain x_2 appropriately so that we get a feasible solution. Whether such a feasible solution will have better objective function value we do not know. *What we do know is, because the objective function value of P_2 (68.8286) is better than the objective function value of P_1 (65), it is worthwhile to check it*. Although the objective function value of successor problems to P_2 can only get worse (lower), they might be better than P_1, and if we can find a successor (or a successor's successor,…) that is feasible, it might be better than our best current feasible solution, which is P_1.

On the other hand, if P_2 had resulted in an objective function lower than that of P_1, what would we have done?

➔ We would not have evaluated any more successor problems to P_2. Why? Because successor nodes add constraints, and it is impossible for the objective to get better by adding constraints. Either the objective will get worse, or it will stay the same. But it

will not get better. Therefore if a predecessor node is already not as good as the current best feasible solution, there is no way a successor node will be, and we might as well terminate evaluation of successors to that predecessor node.

But in this case, the objective function value of P_2 is better than that of our current best feasible solution, so let's solve P_2 with the additional constraint $x_2 \leq 2$. Call this problem P_3, given below.

$$
\begin{aligned}
& \max \quad \zeta = 17x_1 + 12x_2 \\
& \text{s.t.} \\
P_3 \qquad & 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 \\
& x_1 \geq 2 \\
& x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Using CPLEX to solve P_3 results in:

$P_3 \quad Solution: \qquad x_1 = 2.6, x_2 = 2.0, \qquad \zeta = 68.2$

Note that x_1 has reverted back to non-integer. We could have expected this since we forced x_2 to change, requiring a new corner point, as shown in Fig. 18.

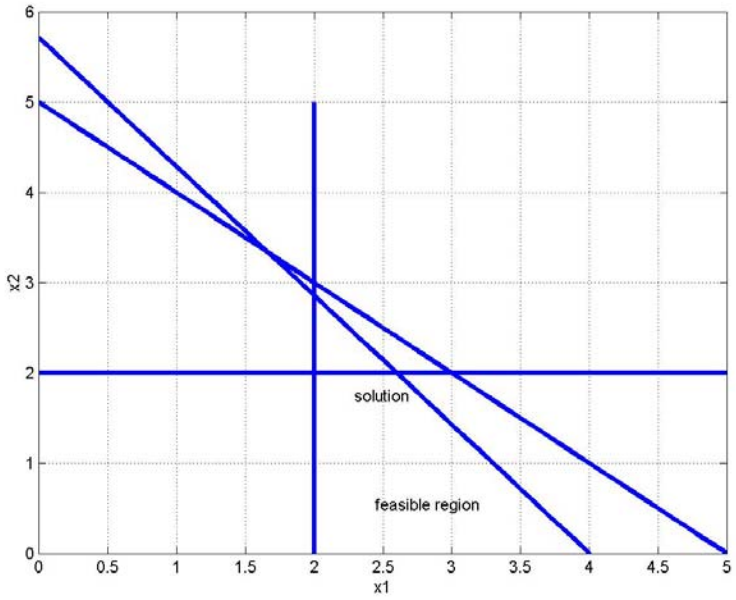


Fig. 18

Now we need to force a feasible solution, and we do so by imposing $x_1 \leq 2$, as indicated in problem P_4 below.

$$
\begin{aligned}
& \max \quad \zeta = 17x_1 + 12x_2 \\
& \text{s.t.} \\
P_4 \qquad & 10x_1 + 7x_2 \leq 40 \\
& x_1 + x_2 \leq 5 \\
& x_1 \leq 2 \\
& x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Using CPLEX to solve P_4, we obtain:

$P_4 \quad Solution: \qquad x_1 = 2.0, x_2 = 2.0, \qquad \zeta = 58.0$

This solution is displayed in Fig. 19.

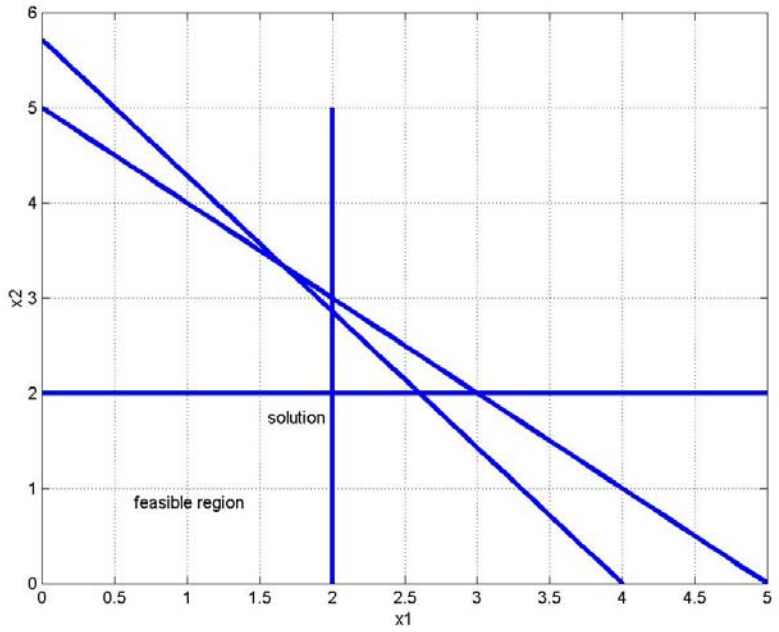


Fig. 19

This solution is feasible! Wonderful. However, comparing the objective value function to our "best so-far" value of 65, we see that this value, 58, is worse. So the P_4 solution is not of interest to us since we already have one that is better.

What next?

Let's go back to problem P_3 where we had this:

$$P_3 \quad Solution: \qquad x_1 = 2.6, x_2 = 2.0, \qquad \zeta = 68.2$$

In P_4, we pushed x_1 to 2. Let's explore the space associated with pushing x_1 to 3. So we need a new problem for this, where we require $x_1 \geq 3$. Let's call this P_5, given below.

$$
\begin{aligned}
&\max \quad \zeta = 17x_1 + 12x_2 \\
&\text{s.t.} \\
&\qquad 10x_1 + 7x_2 \le 40 \\
P_5 \qquad &\qquad x_1 + x_2 \le 5 \\
&\qquad x_1 \ge 3 \\
&\qquad x_2 \le 2 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

Using CPLEX to solve it, we obtain

$P_5 \quad Solution: \qquad x_1 = 3.0, x_2 = 1.4286, \qquad \zeta = 68.1429$

This solution is not integer, but it has an objective function value better than our current best feasible solution (65), and so let's branch again to a successor node with $x_2 \le 1$. Let's call this P_6, given below.

$$
\begin{aligned}
&\max \quad \zeta = 17x_1 + 12x_2 \\
&\text{s.t.} \\
&\qquad 10x_1 + 7x_2 \le 40 \\
P_6 \qquad &\qquad x_1 + x_2 \le 5 \\
&\qquad x_1 \ge 3 \\
&\qquad x_2 \le 1 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

Using CPLEX to solve it, we obtain

$P_6 \quad Solution: \qquad x_1 = 3.3, x_2 = 1.0, \qquad \zeta = 68.1$

Again, the solution is not integer, but it has an objective function value better than our current best feasible solution (65), and so let's branch again to a successor node with $x_1 \le 3$. Let's call this P_7, given below:

$$
\begin{aligned}
&\max \quad \zeta = 17x_1 + 12x_2 \\
&\text{s.t.} \\
&\qquad 10x_1 + 7x_2 \le 40 \\
P_7 \qquad &\qquad x_1 + x_2 \le 5 \\
&\qquad x_1 \le 3 \\
&\qquad x_2 \le 1 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

Using CPLEX to solve it, we obtain

$P_7 \quad Solution: \qquad x_1 = 3.0, x_2 = 1.0, \qquad \zeta = 63.0$

This solution is feasible! However, comparing the objective value function to our "best so-far" value of 65, we see that this value, 63, is worse. So the P_7 solution is not of interest to us since we already have one that is better.

So now what?

Let's go back to problem P_6 where we had this:

$P_6 \quad Solution: \qquad x_1 = 3.3, x_2 = 1.0, \qquad \zeta = 68.1$

In P_7, we pushed x_1 to 3. Let's explore the space associated with pushing x_1 to 4. So we need a new problem for this, where we require $x_1 \ge 4$. Let's call this P_8, given below.

$$
\begin{aligned}
&\max \quad \zeta = 17x_1 + 12x_2 \\
&\text{s.t.} \\
&\qquad 10x_1 + 7x_2 \le 40 \\
P_8 \qquad &\qquad x_1 + x_2 \le 5 \\
&\qquad x_1 \ge 4 \\
&\qquad x_2 \le 1 \\
&\qquad x_1, x_2 \ge 0
\end{aligned}
$$

Using CPLEX to solve it, we obtain

$P_8 \quad Solution: \qquad x_1 = 4.0, x_2 = 0, \qquad \zeta = 68.0$

This solution is feasible! Not only that, but when we compare the objective value function to our "best so-far" value of 65, we see that this value, 68, is better. So the P_8 solution becomes our new "best solution." And we will use 68 as our bound against which we will test other solutions.

What other solutions do we need to test?

Let's go back to problem P_2 where we had this:

$P_2 \quad Solution: \qquad x_1 = 2.0, x_2 = 2.857, \qquad \zeta = 68.8286$

In P_3, we pushed x_2 to 2. Let's explore the space associated with pushing x_2 to 3. So we need a new problem for this, where we require $x_2 \geq 3$. Let's call this P_9, given below.

$$
\begin{aligned}
& \max \quad \zeta = 17x_1 + 12x_2 \\
& \text{s.t.} \\
& \qquad 10x_1 + 7x_2 \leq 40 \\
P_9 \qquad & \qquad x_1 + x_2 \leq 5 \\
& \qquad x_1 \geq 2 \\
& \qquad x_2 \geq 3 \\
& \qquad x_1, x_2 \geq 0
\end{aligned}
$$

Using CPLEX to solve, we learn that Problem P_9 is infeasible.

We have one more branch to check…

Let's go back to problem P_5 where we had this:

$P_5 \quad Solution: \qquad x_1 = 3.0, x_2 = 1.4286, \qquad \zeta = 68.1429$

In P_6, we pushed x_2 to 1. Let's explore the space associated with pushing x_2 to 2. So we need a new problem for this, where we require $x_2 \geq 2$. Let's call this P_{10}, given below.

$$\max \quad \zeta = 17x_1 + 12x_2$$

s.t.

$$P_{10} \qquad \begin{aligned} 10x_1 + 7x_2 &\le 40 \\ x_1 + x_2 &\le 5 \\ x_1 &\ge 3 \\ x_2 &\ge 2 \\ x_1, x_2 &\ge 0 \end{aligned}$$

Using CPLEX to solve, we learn that Problem P_{10} is infeasible.

The tree-search is illustrated in Fig. 20 below.

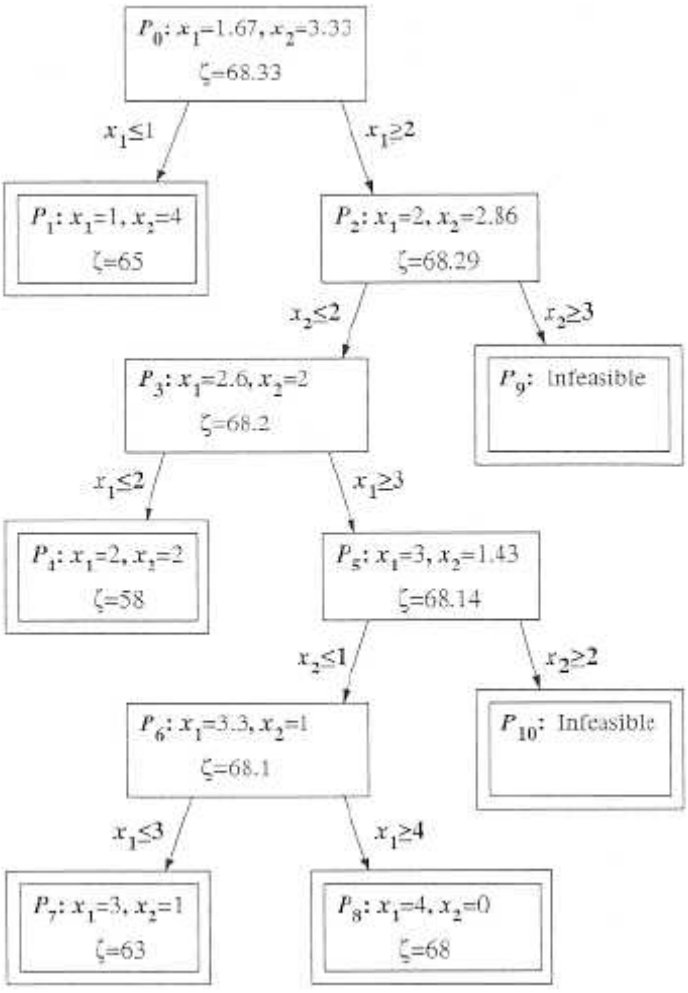


Fig. 20 [13]

There are two central ideas in the B&B method.

1. *Branch*: It uses LP-relaxation to decide how to branch. Each branch will add a constraint to the previous LP-relaxation to enforce integrality on one variable that was not integer in the predecessor solution.
2. *Bound*: It maintains the best integer feasible solution obtained so far as a bound on tree-paths that should still be searched.
    a. If any tree node has an objective value less optimal than the identified bound, no further searching from that node is necessary, since adding constraints can never improve an objective.
    b. If any tree node has an objective value more optimal than the identified bound, then additional searching from that node is necessary.

In the B&B method, the first thing to do is to solve the problem as an LP-relaxation. If the solution results so that all integer variables are indeed integer, the problem is solved.

On the other hand, if the LP-relaxation results in a solution that contains a particular variable $x_k=[x_k]+s_k$, where $0<s_k<1$, and $[x_k]$ is an integer, then we branch to two more linear programs

- The predecessor problem except with the additional constraint: $x_k=[x_k]$
- The predecessor problem except with the additional constraint $x_k=[x_k]+1$

7.4 Other algorithmic issues on B&B

In section 7.3, we identified the essence of the B&B algorithm:

Branch: force integrality on one variable by adding a constraint to an LP-relaxation.

Bound: Continue branching only if the objective function value of the current solution is better than the objective function value of the best feasible solution obtained so far.

It is our primary goal to have understood this.

However, you should be aware that there are several other issues related to B&B that need to be considered before implementation. Below is a brief summary of two of these issues [14, ch. 6].

7.4.1 Node selection

The issue here is this:
➔ Following examination of a node in which we conclude we need to branch, should we go deeper, or should we go broader?

Consider, in the example of Section 4.0, the situation we were in at the node corresponding to Problem P_3. Reference to Fig. 9 above will be useful at this point. The solution we obtained was

$$P_3 \quad Solution: \qquad x_1 = 2.6, x_2 = 2.0, \qquad \zeta = 68.2$$

And so it was clear to us, because our current best feasible solution had an objective function value of 65, and 68.2>65, we needed to branch further on this node.

However, consider P_3's predecessor node, P_2. Its solution was

$$P_2 \quad Solution: \qquad x_1 = 2.0, x_2 = 2.857, \qquad \zeta = 68.8286$$

To get to P_3, we had added the constraint $x_2 \leq 2$. But there was something else we could have done, and that was added the constraint $x_2 \geq 3$. This was the other branch. We had to pick one or the other, and we decided on $x_2 \leq 2$.

But once at P_3, after evaluating it, we had a tough decision to make.

- Depth: Do we continue from P_3, requiring $x_1 \leq 2$, for example? or
- Breadth: Do we go back to P_2 to examine its other branch, $x_2 \geq 3$?

For high-dimensional IPs, it is usually the case that feasible solutions (meaning that all integer variables are integer) are more likely to occur deep in a tree than at nodes near the root. Finding multiple feasible solutions early in B&B is important because it tightens the bound (in the above example, it increases the bound), and therefore enables termination of branching at more nodes (and therefore decreases computation). One can see this by considering the bound before we find a feasible solution: the bound is infinite! ($+\infty$ for maximization problems and $-\infty$ for minimization problems). Therefore the best strategy is to go deep first; then after finding several feasible solutions, explore breadth.

7.4.2 Branching variable selection

In the example of Section 4.0, branching variable selection was not an issue. At every node, there was only, at most, one non-integer variable, and so there was no decision to be made. This was because our example problem had only two variables, x_1 and x_2.

However, if we consider having any more than 2 decision variables, even 3, we run into this issue, and that is:
➔ Given we are at a node that needs to branch, and there are 2 or more non-integer variables, which one do we branch on?

This is actually a rich research question that, so far, has not been solved generically but rather, is addressed for individual problem types by pre-specifying an *ordering* of the variables that are required to be integer. Good orderings become apparent after running the algorithm many times for many different conditions. Sometimes, good orderings are apparent based on some physical understanding of the problem, e.g., perhaps the largest unit should be chosen first.

7.4.3 Mixed integer problems

Our example focused on a pure-integer program (PIP). However, the UC is a mixed integer problem (MIP). Do we need to consider this more deeply?

Reconsider our example problem, except now consider it as a mixed integer problem as follows:

$$
\begin{aligned}
& \max \quad \zeta = 17x_1 + 12x_2 \\
& \text{s.t.} \\
IP_2 \qquad & 10x_1 + 7x_2 \le 40 \\
& x_1 + x_2 \le 5 \\
& x_1, x_2 \ge 0 \\
& x_1 \text{ integer.}
\end{aligned}
$$

Recall our tree-search diagram, repeated here for convenience. You should be able to see that the solution is obtained as soon as we solved P_1 and P_2.

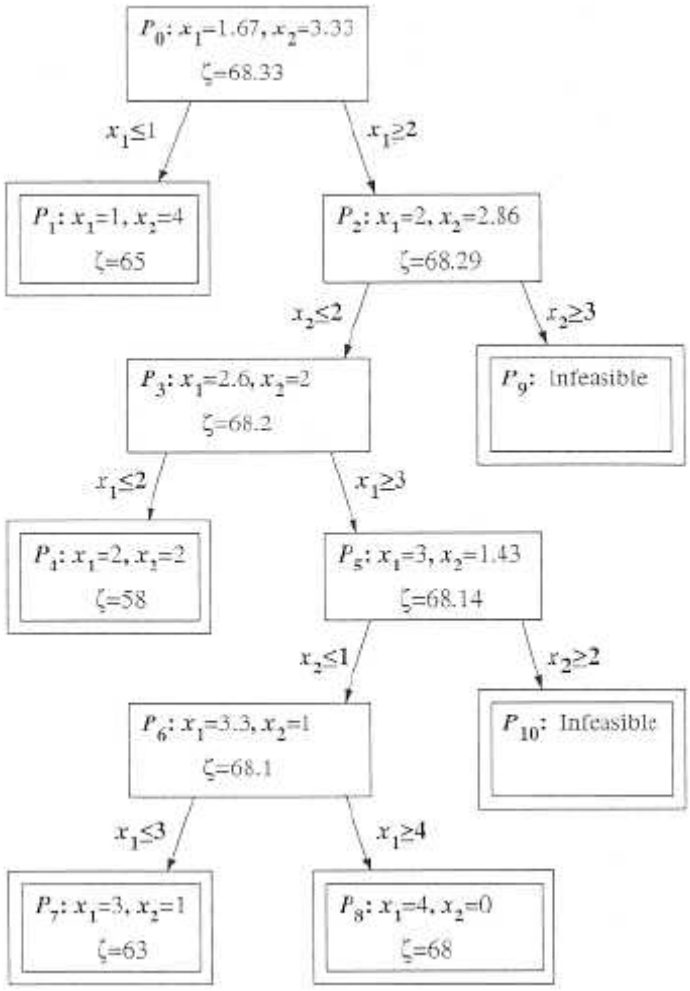


Fig. 21 [13]

The other possible MIP would be to require x_2 integer only, as follows:

$$\max \quad \zeta = 17x_1 + 12x_2$$

s.t.

$$IP_3 \qquad \begin{aligned} 10x_1 + 7x_2 &\le 40 \\ x_1 + x_2 &\le 5 \\ x_1, x_2 &\ge 0 \\ x_2 &\text{ integer.} \end{aligned}$$

Recalling the solution to P_0,

$$P_0 \quad Solution: \qquad x_1 = 1.667, x_2 = 3.333, \qquad \zeta = 68.333$$

we see that we can branch by setting either $x_2 \leq 3$ or $x_2 \geq 4$, resulting in

$x_2 \leq 3$:

$$Solution: \quad x_1 = 1.9, x_2 = 3.0, \qquad \zeta = 68.3$$

$x_2 \geq 4$:

$$Solution: \quad x_1 = 1.0, x_2 = 4.0, \qquad \zeta = 65$$

And so we know the solution is the one for $x_2 \leq 3$.

The conclusion here is that we can very easily solve MIP within our LP-relaxation branch and bound scheme by simply allowing the non-integer variables to remain relaxed.

7.5 Using CPLEX to solve MIPs directly

Here is the sequence I used to call CPLEX's MIP-solver to solve our example problem above.

1. Created the problem statement within a text file called mip.lp as follows:

```
maximize
 17 x1 + 12 x2
subject to
 10 x1 + 7 x2 <= 40
    x1 +   x2 <= 5
Bounds
 0<= x1 <= 1000
 0<= x2 <= 1000
Integer
 x1 x2
end
```

3. Used WinSCP to port the file to pluto.

4. Used Putty to log on to pluto.
5. Typed cplex101 to call cplex.
6. Typed read mip.lp to read in problem statement.
7. Typed mipopt to call the MIP-solver. The result was as follows:

```
Tried aggregator 1 time.
MIP Presolve modified 3 coefficients.
Reduced MIP has 2 rows, 2 columns, and 4 nonzeros.
Presolve time =    0.00 sec.
MIP emphasis: balance optimality and feasibility.
Root relaxation solution time =    0.00 sec.

      Nodes                                         Cuts/
 Node  Left     Objective  IInf  Best Integer    Best Node    ItCnt     Gap

    0     0       68.3333     2                68.3333        2
*    0+    0                          0.0000       68.3333        2    5.13%
*                68.0000     0       68.0000      Cuts:  3        4    0.00%

Mixed integer rounding cuts applied:  1

MIP - Integer optimal solution:  Objective =  6.8000000000e+01
Solution time =    0.00 sec.  Iterations = 4  Nodes = 0
```

8. Typed display solution variables - to display solution variables. The result was:

```
Variable Name           Solution Value
x1                            4.000000
All other variables in the range 1-2 are 0.
```

8.0 UC Solution by mixed-integer programming

Here, we provide some data to use in solving our problem.
We illustrate using an example that utilizes the same system we have been using in our previous notes, where we had 3 generator buses in a 4 bus network supplying load at 2 different buses, but this

time we will model each generator with the ability to submit 3 offers.

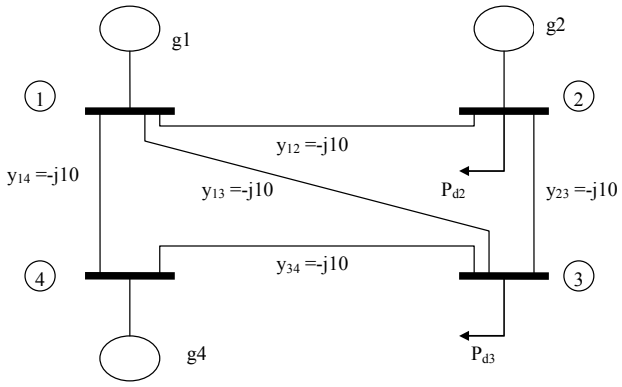


Fig. 22: One line diagram for example system

The offers, in terms of fixed costs, production costs, and corresponding min and max generation limits are as follows

Production costs (in $/pu-hr):

Unit, k	Fixed costs ($/hr)	Startup Costs ($)	Shutdown Costs ($)	Production Costs ($/pu-hr)		
				g_{k1t}	g_{k2t}	g_{k4t}
1	50	100	20	1246	1307	1358
2	50	100	20	1129	1211	1282
4	50	100	20	1183	1254	1320

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} g_{11t} \\ g_{12t} \\ g_{13t} \\ g_{21t} \\ g_{22t} \\ g_{23t} \\ g_{41t} \\ g_{42t} \\ g_{43t} \end{bmatrix} \leq \begin{bmatrix} 0.50 \\ 0.60 \\ 0.40 \\ 0.35 \\ 0.60 \\ 0.20 \\ 0.45 \\ 0.50 \\ 0.40 \end{bmatrix}, \forall t$$

The UC problem is for a 24 hour period, with loading data given as below. Figure 2, the load curve, illustrates variation of load with time over the 24 hour period.

Hour, t	Load, D_t (pu)
1	1.50
2	1.40
3	1.30
4	1.40
5	1.70
6	2.00
7	2.40
8	2.80
9	3.20
10	3.30
11	3.30
12	3.20
13	3.20
14	3.30
15	3.35
16	3.40
17	3.30
18	3.30
19	3.20
20	2.80
21	2.30
22	2.00
23	1.70
24	1.60

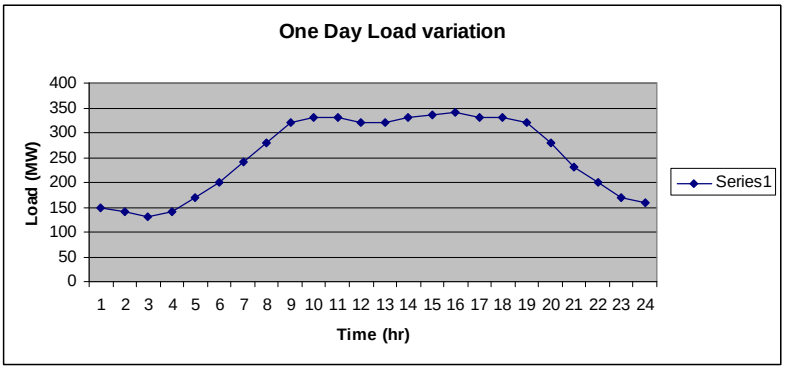


Fig. 23: Load curve

8.1 Example – 4 hours

For this solution, we will only include startup and shutdown constraints. In order to illustrate all data entered, we will analyze only the first four hours. The CPLEX code to do this is given below.

```
minimize
 50 z11 +50 z12 + 50 z13 + 50 z14
+50 z21 +50 z22 + 50 z23 +50 z24
+50 z41 +50 z42 + 50 z43 +50 z44
+1246 g111 + 1307 g121 + 1358 g131
 +1129 g211 + 1211 g221 + 1282 g231
 +1183 g411 + 1254 g421 + 1320 g431
+1246 g112 + 1307 g122 + 1358 g132
 +1129 g212 + 1211 g222 + 1282 g232
 +1183 g412 + 1254 g422 + 1320 g432
+1246 g113 + 1307 g123 + 1358 g133
 +1129 g213 + 1211 g223 + 1282 g233
 +1183 g413 + 1254 g423 + 1320 g433
+1246 g114 + 1307 g124 + 1358 g134
 +1129 g214 + 1211 g224 + 1282 g234
 +1183 g414 + 1254 g424 + 1320 g434
+100 y12 + 100 y13 +100 y14
+100 y22 + 100 y23 +100 y24
+100 y42 + 100 y43 +100 y44
+20 x12 + 20 x13 +20 x14
+20 x22 + 20 x23 + 20 x24
+20 x42 + 20 x43 +20 x44
subject to
 loadhr1: g111+g121+g131+g211+g221+g231+g411+g421+g431=1.5
 loadhr2: g112+g122+g132+g212+g222+g232+g412+g422+g432=1.4
 loadhr3: g113+g123+g133+g213+g223+g233+g413+g423+g433=1.3
 loadhr4: g114+g124+g134+g214+g224+g234+g414+g424+g434=1.4
 initialu1: z11=0
 initialu2: z21=1
 initialu4: z41=1
 starthr21u1: z12-z11-y12<=0
 starthr32u1: z13-z12-y13<=0
 starthr43u1: z14-z13-y14<=0
 starthr21u2: z22-z21-y22<=0
 starthr32u2: z23-z22-y23<=0
 starthr43u2: z24-z23-y24<=0
 starthr21u4: z42-z41-y42<=0
 starthr32u4: z43-z42-y43<=0
 starthr43u4: z44-z43-y44<=0
 shuthr21u1: z12-z11+x12>=0
 shuthr32u1: z13-z12+x13>=0
 shuthr43u1: z14-z13+x14>=0
 shuthr21u2: z22-z21+x22>=0
 shuthr32u2: z23-z22+x23>=0
 shuthr43u2: z24-z23+x24>=0
 shuthr21u4: z42-z41+x42>=0
 shuthr32u4: z43-z42+x43>=0
 shuthr43u4: z44-z43+x44>=0
```

g111 - 0.5 z11<= 0
g112 - 0.5 z12<= 0
g113 - 0.5 z13<= 0
g114 - 0.5 z14<= 0
g121 - 0.6 z11<= 0
g122 - 0.6 z12<= 0
g123 - 0.6 z13<= 0
g124 - 0.6 z14<= 0
g131 - 0.4 z11<= 0
g132 - 0.4 z12<= 0
g133 - 0.4 z13<= 0
g134 - 0.4 z14<= 0

g211 - 0.35 z21<= 0
g212 - 0.35 z22<= 0
g213 - 0.35 z23<= 0
g214 - 0.35 z24<= 0
g221 - 0.6 z21<= 0
g222 - 0.6 z22<= 0
g223 - 0.6 z23<= 0
g224 - 0.6 z24<= 0
g231 - 0.2 z21<= 0
g232 - 0.2 z22<= 0
g233 - 0.2 z23<= 0
g234 - 0.2 z24<= 0

g411 - 0.45 z41<= 0
g412 - 0.45 z42<= 0
g413 - 0.45 z43<= 0
g414 - 0.45 z44<= 0
g421 - 0.5 z41<= 0
g422 - 0.5 z42<= 0
g423 - 0.5 z43<= 0
g424 - 0.5 z44<= 0
g431 - 0.4 z41<= 0
g432 - 0.4 z42<= 0
g433 - 0.4 z43<= 0
g434 - 0.4 z44<= 0

```
Bounds
 0<= g111
 0<= g112
 0<= g113
 0<= g114
 0<= g121
 0<= g122
 0<= g123
 0<= g124
 0<= g131
 0<= g132
 0<= g133
 0<= g134

 0<= g211
 0<= g212
 0<= g213
 0<= g214
 0<= g221
 0<= g222
 0<= g223
 0<= g224
 0<= g231
 0<= g232
 0<= g233
 0<= g234

 0<= g411
 0<= g412
 0<= g413
 0<= g414
 0<= g421
 0<= g422
 0<= g423
 0<= g424
 0<= g431
 0<= g432
 0<= g433
 0<= g434
Integer
 z11 z12 z13 z14
 z21 z22 z23 z24
 z41 z42 z43 z44
 y12 y13 y14
 y22 y23 y24
 y42 y43 y44
 x12 x13 x14
 x22 x23 x24
 x42 x43 x44
end
```

Result: CPLEX gives an objective function value of 7020.7 $.

CPLEX> display solution variables -

Variable Name	Solution Value
z21	1.000000
z22	1.000000
z23	1.000000
z24	1.000000
z41	1.000000
z42	1.000000
z43	1.000000
z44	1.000000
g211	0.350000
g221	0.600000
g411	0.450000
g421	0.100000
g212	0.350000
g222	0.600000
g412	0.450000
g213	0.350000
g223	0.500000
g413	0.450000
g214	0.350000
g224	0.600000
g414	0.450000

All other variables in the range 1-66 are 0.

Note that all y- and x-variables are zero, therefore there is no starting up or shutting down.
One should check that the generation in each hour equals the demand in that hour:
g211+g221+g411+g421=0.35+0.6+0.45+0.1=1.5
g212+g222+g412=0.35+0.6+0.45=1.4
g213+g223+g413=0.35+0.5+0.45=1.3
g214+g224+g414=0.35+0.6+0.45=1.4

This very simple solution was obtained as a result of the fact that the initial solution of

initialu1: z11=0
initialu2: z21=1
initialu4: z41=1

was in fact the best one for the initial loading condition, and since the loading condition hardly changed during the first four hours, there was no reason to change any of the units.

Let's try a different initial condition:

initialu1: z11=1
initialu2: z21=0
initialu4: z41=1

Result: CPLEX gives an objective function value of 7208.9 $.

CPLEX> display solution variables -

Variable Name	Solution Value
z11	1.000000
z22	1.000000
z23	1.000000
z24	1.000000
z41	1.000000
z42	1.000000
z43	1.000000
z44	1.000000
g111	0.500000
g121	0.050000
g411	0.450000
g421	0.500000
g212	0.350000
g222	0.600000
g412	0.450000
g213	0.350000
g223	0.500000
g413	0.450000

g214	0.350000
g224	0.600000
g414	0.450000
y22	1.000000
x12	1.000000

All other variables in the range 1-66 are 0.
Why was this solution more expensive?

➔ Because we initialized the solution with more expensive units, to get back to the less expensive solution, notice that the program forces unit 2 to start up (y_{22}=1) and unit 1 to shut down (x_{12}=1) at the beginning of period 2. Apparently, the additional cost of starting unit 2 ($100) and shutting down unit 1 ($20) was not enough to offset the savings associated with running the more efficient unit over the remaining three hours of the simulation.

Let's test our theory by increasing the startup costs of unit 2 from $100 to $10,000. The objective function value in this case is $7281.25 (higher than the last solution). The decision variables are:

Variable Name	Solution Value
z11	1.000000
z12	1.000000
z13	1.000000
z14	1.000000
z41	1.000000
z42	1.000000
z43	1.000000
z44	1.000000
g111	0.500000
g121	0.050000
g411	0.450000
g421	0.500000
g112	0.500000
g412	0.450000
g422	0.450000

g113	0.500000
g413	0.450000
g423	0.350000
g114	0.500000
g414	0.450000
g424	0.450000

All other variables in the range 1-66 are 0.

We observe that unit 1 was on-line the entire four hours, i.e, there was no switching, something we expected since the start-up cost of unit 2 was so very high.

8.2 Example – 24 hours

We refrain from providing the data in this case because it is extensive, having 426 variables:

72 z-variables
69 y-variables
69 x-variables
216 g-variables

Rather, we have posted the dataset on the web page under "UC24 Data" for the 04/16/08 date.

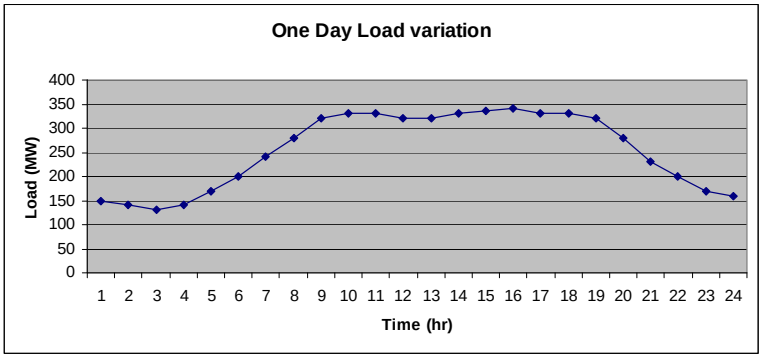
The solution was initialized at

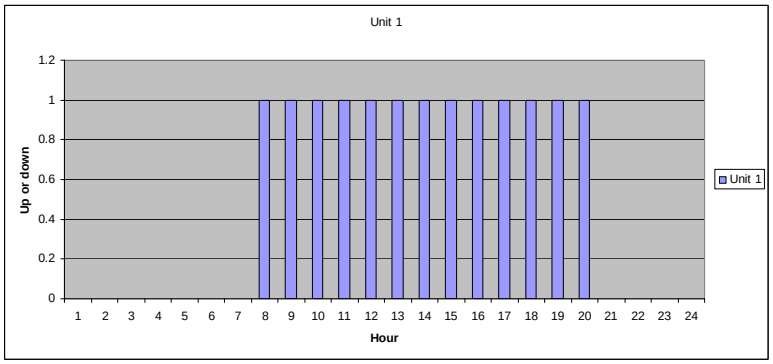
initialu1: z11=0
initialu2: z21=1
initialu4: z41=1

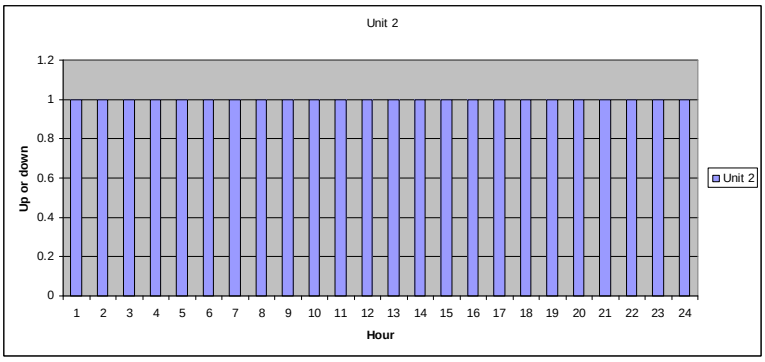
which is the most economic solution for this loading level.

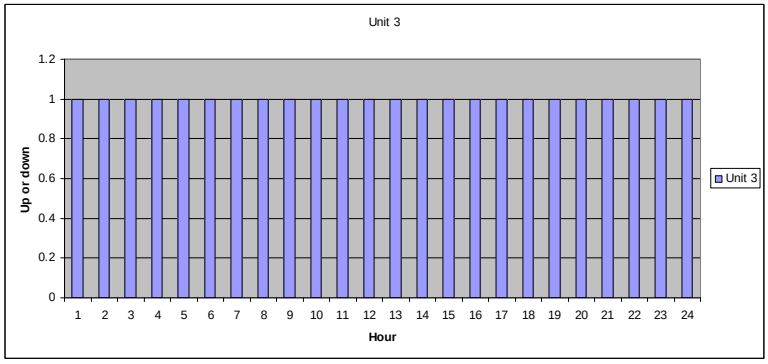
The output is most easily analyzed by using
"display solution variables -"
and then searching the output variables for y-variables and/or x-variables that are listed (and therefore 1). These variables indicate changes in the unit commitment. In studying the load curve, what kind of changes do you expect?

The result, objective value=$77667.3, shows that the only x and y variables that are non-zero are y18 and x121. This means that the changes in the unit commitment occur only for unit 1 and only at hours 18 and 21. A pictorial representation of the unit commitment through the 24 hour period is shown below.

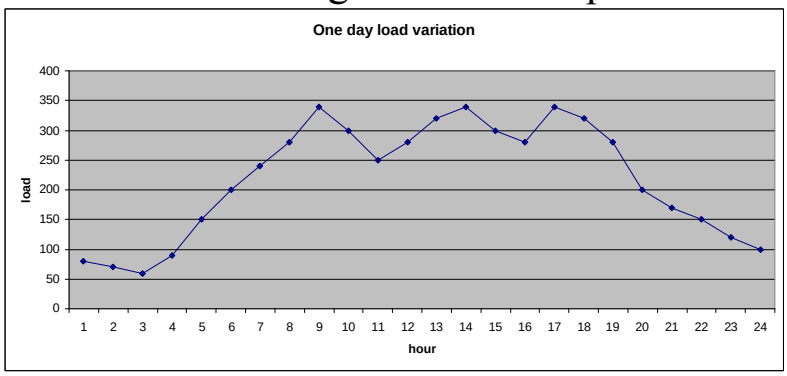


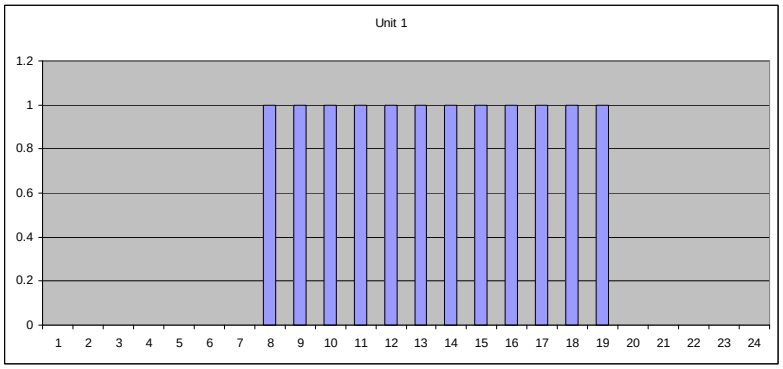


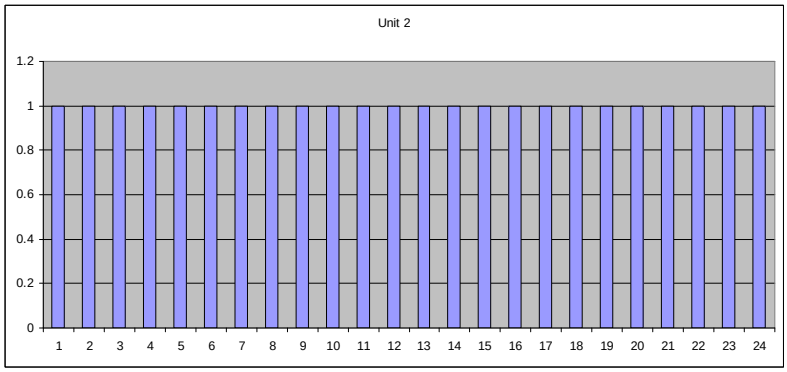


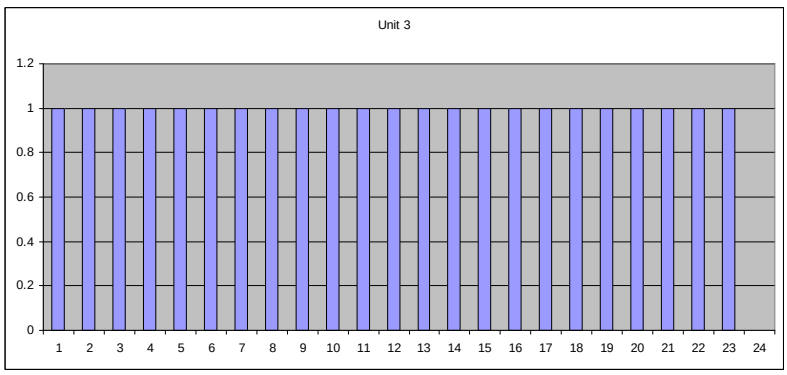


To perform additional investigation, the load curve was modified as shown below (UC24a.lp). All other data remained as before. The result, with objective function value of $, shows that the only x and y variables that are non-zero are y18, x120, and x424. A pictorial representation of the UC through the 24 hour period is shown below.

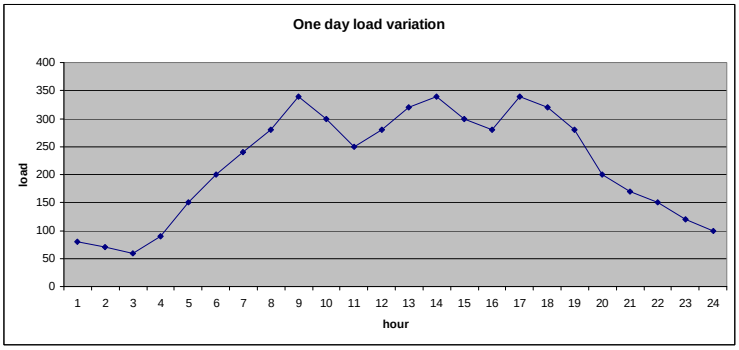


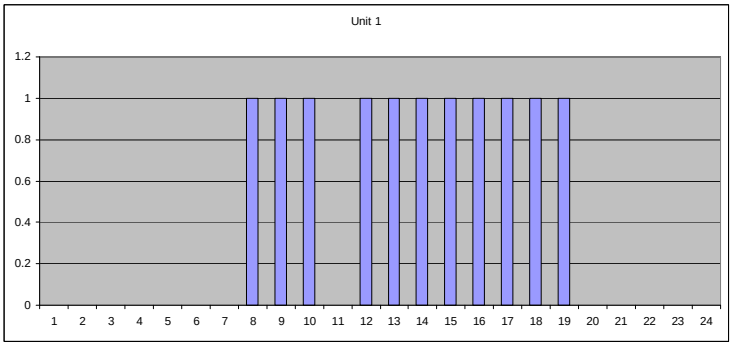


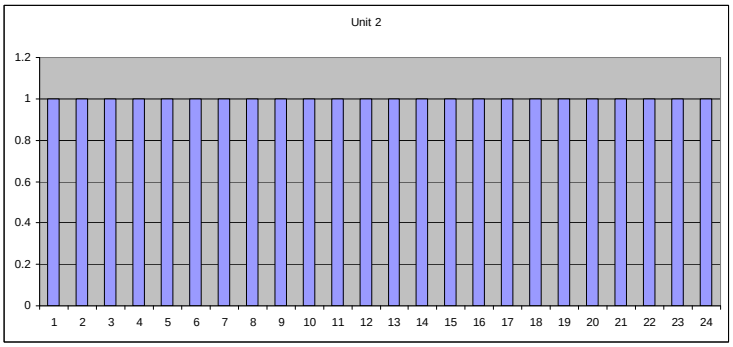


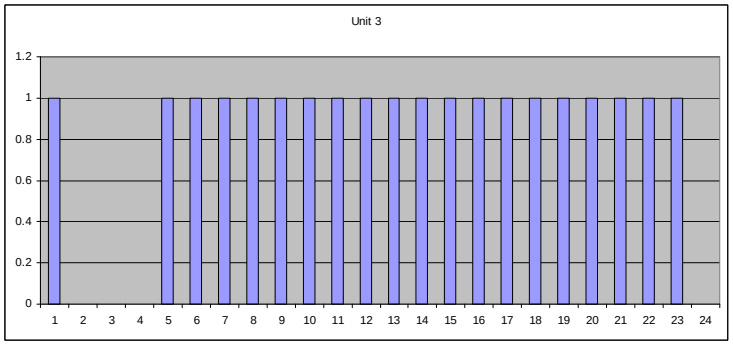


In a last investigation, the load curve remained modified, and startup costs were reduced to $10, shutdown costs reduced to $2. All other data remained as before (UC24b.lp). The result, with objective function value of $66,867.95, shows that the only x and y variables that are non-0 are y18, x112, y45, x111, x120, x42, x424. A pictorial representation of the UC through the 24 hour period is shown below.









[1] "PJM Emergency Procedures," www.pjm.com/etools/downloads/edart/edart-training-pres/edart-training-instantaneous-revserve-check.pdf.
[2] H. Pinto, F. Magnago, S. Brignone, O. Alsaç, B. Stott, "Security Constrained Unit Commitment: Network Modeling and Solution Issues," Proc. of the 2006 IEEE PES Power Systems Conference and Exposition, Oct. 29 2006-Nov. 1 2006, pp. 1759 – 1766.
[3] R. Chhetri, B. Venkatesh, E. Hill, "Security Constraints Unit Commitment for a Multi-Regional Electricity Market," Proc. of the 2006 Large Engineering Systems Conference on Power Engineering, July 2006, pp. 47 – 52.
[4] J. Guy, "Security Constrained Unit Commitment," IEEE Transactions on Power Apparatus and Systems Vol. PAS-90, Issue 3, May 1971, pp. 1385-1390.
[5] B. Hobbs, M. Rothkopf, R. O'Neill, and H. Chao, editors, "The Next Generation of Electric Power Unit Commitment Models," Kluwer, 2001.
[6] M. Rothleder, presentation to the Harvard Energy Policy Group, Dec 7, 2007.
[7] J. Chow, R. De Mello, K. Cheung, "Electricity Market Design: An Integrated Approach to Reliability Assurance," Proceedings of the IEEE, Vol. 93, No. 11, November 2005.
[8] Q. Zhou, D. Lamb, R. Frowd, E. Ledesma, A. Papalexopoulos, "Minimizing Market Operation Costs Using A Security-Constrained Unit Commitment Approach," 2005 IEEE/PES Transmission and Distribution Conference & Exhibition: Asia and Pacific Dalian, China.
[9] A. Ott, "Experience with PJM Market Operation, System Design, and Implementation," IEEE Transactions on Power Systems, Vol. 18, No. 2, May 2003, pp. 528-534.
[10] D. Streiffert, R. Philbrick, and A. Ott, "A Mixed Integer Programming Solution for Market Clearing and Reliability Analysis," Power Engineering Society General Meeting, 2005. IEEE 12-16 June 2005 , pp. 2724 - 2731 Vol. 3.
[11] F. Hillier and G. Lieberman, "Introduction to Operations Research," fourth edition, Holden-Day, 1986.
[12] T. Hu, "Integer Programming and Network Flows," Addison-Wesley, 1970.
[13] R. Vanderbei, "Linear Programming, Foundations and Extensions," third edition, Springer, 2008.

[14] G. Nemhauser, A. Kan, and M. Todd, editors, “Handbook in Operations Research and Management Science, Volume 1: Optimization,” North Holland, Amsterdam, 1989.