OPTIMIZATION FRAMEWORK FOR NATURAL GAS STORAGE ASSET OPERATION

By

Saurabh Suresh Chalke

Submitted in partial fulfillment of the requirements
for the degree of Master of Engineering

At

Dalhousie University
Halifax, Nova Scotia
December 2019

Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

NG	Natural Gas
LP	Linear Programming
SLP	Stochastic Linear Programming
DDP	Deterministic Dynamic Programming
MILP	Mixed Integer Linear Programming
RO	Robust Optimization
MMR	Minimizing the Maximum Regret

ABSTRACT

Natural Gas (NG) offers wide range of applications from power generation to room heating. North America has a very complex NG supply chain network comprising of exploration and processing facilities, pipeline transportation, local distribution, and storage facilities. NG being a trading commodity makes its price subject to uncertainty due to supply-demand market dynamics. Companies at the NG storage asset level can make profit by injecting the NG when the prices are low and selling it when they become high. This study presents the implementation of a multiple mathematical programming models to optimize the NG storage asset operation. The study begins with the formulation of the problem as a stochastic program, then use of K-means clustering algorithm over price data to reduce the number of price scenarios (clusters). We also proposed Min-Max Regret (MMR), Robust Optimization (RO), and Deterministic Dynamic Programming (DDP) models. Stochastic LP model is then further studied with the incorporation of chance constraint to ensure some probabilistic guarantee of profit level. The optimization models provide decision support for NG storage assets. In addition, the study also presents a glimpse of Graphic User Interface tool developed for the industry client.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Dr. Ahmed Saif for giving me vision, continuous encouragement, and expert guidance throughout this research. This project would not have been possible without his advice, patience and invaluable comments in all stages of the work. Special thanks to Dr. Uday Venkatadri, for his valuable time and suggestions given to me in my courses and projects.

My sincerest appreciation to all my teachers and Industrial Engineering staff, who have always helped me with their expert guidance in my studies. Special appreciation is due to my parents for encouraging and believing in me. Finally, I would like to thank all the graduate students in the Department of Industrial Engineering at Dalhousie University and wish them all the best.

Chapter 1: Introduction

Natural gas (NG) is a fossil energy source which contains methane as a principal constituent. It is widely used in many capacities in North America which includes residential heating, commercial/industrial purposes (eg. waste treatment and power generation), etc. According to the NG stats report issued by Canadian Gas Association on January'2018, NG meets over 30% of Canada's energy demand and NG user can save an average of US$1,619 a year compared to other energy options. NG is among the efficient, affordable and reliable energy sources [1].

Starting from the point of exploration of NG from beneath the earth surface, it undergoes extensively complex elements of NG Supply Chain (SC) network such as pre-processing, transportation, storage, and then distribution until it reaches to the customers [1]. NG storage plays a vital role among all other elements as it enables the activation of upstream and downstream elements of NG SC network (exploration and transportation facilities at the upstream level activates during the injection and on the other hand, distribution facility gets activated during withdrawal operation of NG storage asset) [5]. Also, average variability in consumption of NG is much greater than the average variability in production [13].

Although storage facility is the key element of NG-SC, it is difficult to evaluate the NG storage contract due to stochastic nature of the NG prices. However, it is still possible to create a model framework that examines the quantity of natural gas to be injected and/or withdrawn over a given time horizon and forecasted price data, considering the operational constraints and characteristics of NG storage facility[5].

Extensive research has been conducted by researchers over NG supply chain optimization problems for several decades. Avery et al. focused on Local Distribution Companies' (LDC) operations and found that the purchased gas cost accounts for over 60% of their total cost.

Therefore, over a short-term contract with the customers, LDC's primary goal is to supply the NG at minimum cost to meet the expected demand. On the other hand, over a long-term contract, the objective is to maintain a balanced portfolio of different potential supply sources such as pipeline, storage, and transportation of NG directly from producers. The optimization model deals with meeting variable demand through different NG sources [2]. Guldmann & Wang, however, focus on optimal selection of supply mix from different sources to LDCs considering the short term (Weather) demand variability factors [6]. The demand variability due to weather has been considered based on the concept of daily heating degree days.

The gas flow model under demand uncertainty proposed by Levary and Dean [7] considers future demand variability based on past track record of heating degree days and extrapolating the past track records for future forecasts to minimize the expected shortage of NG. Bopp et al. [8] developed a single objective optimization model that explicitly considers the demand uncertainty and deliverability very efficiently as it further extends the concept of heating degree days by dividing a typical year based on probability of occurrence of winter severity (warm winter, average winter, cold winter, and very cold winter) from past data of 20 years of recorded heating degree days in a particular region. The authors have considered the deliverability in terms of severity of winter.

Deliverability refers to the rate at which the reservoir can release gas from its reserves. Deliverability is highest when the reservoir is closest to its maximum capacity and lowest when the reservoir is nearly empty. The injection capacity describes the rate at which natural gas can be pumped into storage for later use. In contrast to the deliverability, the injection rate is lowest when the reservoir is at maximum capacity and highest when the reservoir is empty [4]. Depending on the system configuration and the price model used, approximately between 50% and 62% of the

value of storage can be attributed to natural gas volatility (stochastic variability), with the remaining part attributable to price seasonality (deterministic variability) [19].

The data of real-world optimization problems more often than not are uncertain, i.e., not known exactly at the time the problem is being solved. The reasons for data uncertainty include, among others: measurement/estimation errors coming from the impossibility to measure/estimate exactly the data entries representing characteristics of physical systems/technological processes/ environmental conditions, etc. [12]. Explicitly including uncertainty in a quantitatively structured manner helps improve NG procurement analysis. Two areas of uncertainty complicate the problem: the uncertainty in fuel requirements (demand) and the uncertainty in future fuel prices and availability (supply) [14].

Since there are often many price scenarios to consider, and an array of operational constraints surrounding the storage facility, optimizing the storage asset analytically can require a lot of time and effort. Optimization models are useful when there exist many possible alternatives. Using optimization techniques, we can allow the model to choose the storage times and quantities, such that the expected profit is maximized (according to available data). The models can then be used as decision support tools when utilizing a NG storage asset [9].

The remainder of this project is structured as follows: In Chapter 2, stochastic linear programming (SLP) model based on provided NG price data is presented, along with its results. Next, the price scenarios generation through K-means clustering method and selection of optimal number of price scenarios is discussed in Chapter 3, followed by reformulation of SLP model incorporating the probabilities of optimal price scenarios. Chapter 4 deals with experimenting the SLP model using chance constraint, minimizing the maximum regret and robust optimization approaches to account for price uncertainty. Deterministic dynamic programming (DDP) model is

described with its results in Chapter 5. Model summary is then presented in Chapter 6, describing the results of all the models. In Chapter 7, the user interface tool developed for the industry client (company) is presented in brief. Lastly, the conclusions are drawn for all the SLP model variants and DDP model in Chapter 8.

Chapter 2: Stochastic Linear Programming

The first stage of the project considered studying and benchmarking the work carried out by Paul Jobinpicard (MASc. student) in his thesis. The MILP model developed by Paul handled uncertainty simply by using the average of all the available NG price scenarios for every time period. Therefore NG prices for different time periods were assumed completely independent of one another. However, it is always better to use the trajectory of NG prices over a complete-time horizon, since in reality prices in different time periods are dependent. Paul's MILP model features a deterministic formulation since it does not have constraints for every scenario. Hence, the injection, withdrawal, and inventory quantity variables were also not indexed by scenario. Instead, we formulate the SLP problem considering the constraints for every scenario. In the SLP formulation, scenarios have equal probabilities, i.e. $1/|S|$, where $|S|$ is the number of scenarios. Few limitations over NG asset operating characteristics were outlined based on the approval from project members of the company as follows [9]:

1. Storage asset must be returned to its initial state at the end of last time period
2. Storage asset available for 24 months
3. No holding cost
4. Loss of NG during injection/withdrawal and evaporation losses are ignored
5. Injections and withdrawals cannot occur in the same time period

2.1 Model Formulation

To formulate the problem, we use the following sets, parameters, and variables:

Sets:

$s \in S$: Price Simulations, 1, 2, …, $|S|$

$t \in T$: Time Periods, 1, 2, …, $|T|$

Parameters:

p_{st} = Price of NG at time t according to scenarios s

Cap = Reservoir capacity

C_{in} and C_{out} = Max. injection and withdrawal quantity during a single time t, respectively

$1/|S|$ = Scenario probability

I_0 = Inventory at the beginning of first time period and at the end of last time period

Variables:

x_{st} = Quantity of NG to be injected during time t according to scenario s. A negative x_{st} value denotes withdrawal.

I_{st} = Inventory of NG at the beginning of time t according to scenario s

The Stochastic Linear Program is formulated as follows:

$$maximize\ (-1/|S|) * \sum_{s \in S} \sum_{t \in T} p_{st} \,.\, x_{st} \tag{1}$$

$s.t.$

$$-C_{out} \leq x_{st} \leq C_{in} \qquad \forall s \in S, \forall t \in T \tag{2}$$

$$-x_{st} \leq I_{st} \leq Cap \qquad \forall s \in S, \forall t \in T \tag{3}$$

$$I_{st} = I_{s(t-1)} + x_{s(t-1)} \qquad \forall s \in S, \forall t \in T \tag{4}$$

$$I_{s0} = I_0 \tag{5}$$

$$I_{s|T|} + x_{s|T|} = I_0. \tag{6}$$

The objective function (1) maximizes the expected profit value considering $|S|$ price scenarios with probability $1/|S|$ for each scenario. The negative sign is added to account for injection quantities (positive x_{st} values) and withdrawal quantities (negative x_{st} values). Constraint (2)

ensures the deliverability (maximum injection and withdrawal quantities) of NG storage facility for time t and scenario s. Constraint (3) ensures the capacity of storage facility is obeyed and limits the maximum withdrawal quantity to the available inventory of NG. Constraint (4) ensures the flow balance. Constraint (5) and (6) ensures the NG storage facility returns to its original state by setting both the equations to some value I_0.

2.2 Results

The above formulation was implemented using the following operating characteristics:

Table 1: Operating Characteristics [9]

C_{in}	305,000 MMBtu/month
C_{out}	457,500 MMBtu/month
I_0	0
Cap	1,000,000 MMBtu
$\|T\|$	24 months
$\|S\|$	962

Using the provided data, the following results and policy decision profile were obtained:

Table 2: SLP model results

Expected Profit:	US$ 535,198
Computation Time in Seconds:	11.89

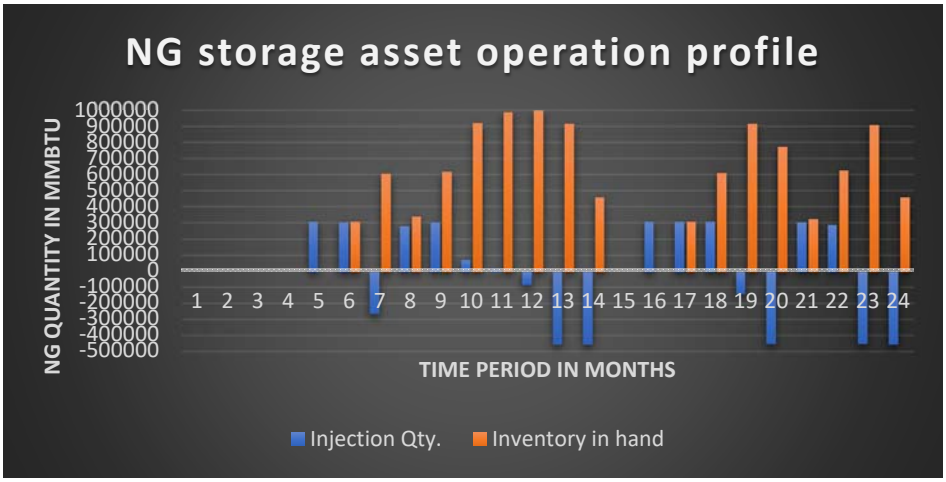


Figure 1: NG storage asset operation profile for SLP model

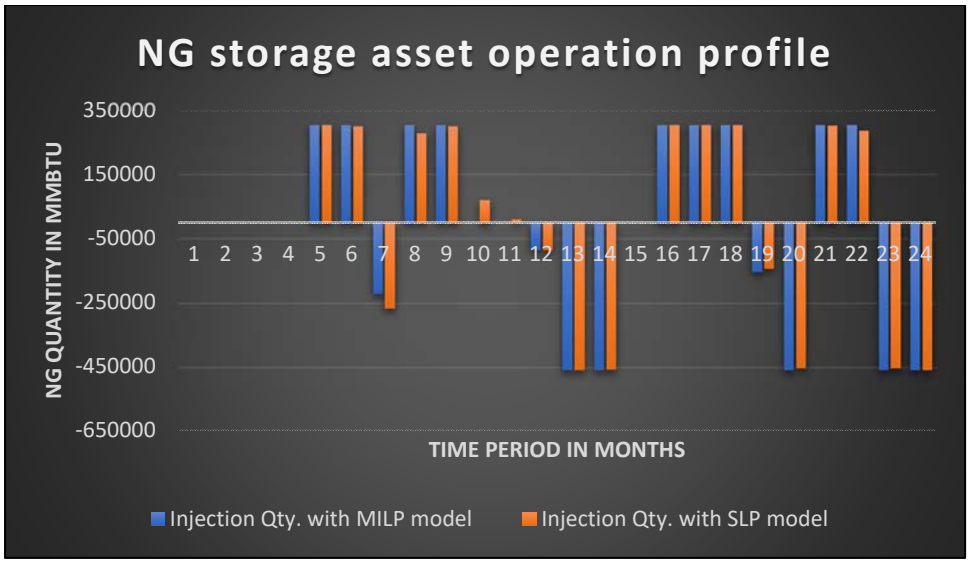


Figure 2: NG storage asset operation profile for MILP & SLP model

We can see from the Table 2 that the SLP model gives the expected profit value $536,198, which is around 0.1% greater than the output of Paul's MILP model. Since we considered the constraints for every scenario, the SLP model took few extra seconds to get the output. We can see from Figure 1 that all the constraints (2 to 6) are completely satisfied. Figure 2 shows the comparison of NG storage operation profile for MILP and SLP model. MILP model gives 17 storage operations, whereas SLP gives two additional storage operations during time period 10 & 11. Apart from that, both the profiles appears to be almost similar with little deviation in injection quantities during time period 7, 8, 19, 20 & 22.

Chapter 3: Stochastic Linear Programming with small number of price scenarios

In the previous modeling approach, the large number of price scenarios makes the problem difficult to solve, we try to generate a manageable number of path scenarios using clustering algorithm. K-means clustering method was implemented to get the definite price scenarios (vectors).

3.1 Price scenarios generation through K-means clustering algorithm

The main idea behind the K-means clustering algorithm is to obtain the definite price scenarios for which the within-cluster variation is as small as possible. The algorithm follows certain steps as listed below [15]:

Step 1: Randomly assign a number, from 1 to K, to each of the price scenarios to get the initial cluster assignment.

Step 2: Calculate the mean for all the clusters.

Step 3: Re-assign the observations to the nearest mean.

Step 4: Iterate the step 2 and 3 until the cluster assignments stop changing.

The K-means clustering problem can be mathematically expressed as:

$$\underset{C_1,\ldots,C_K}{minimize} \left\{ \sum_{k=1}^{K} W(C_k) \right\}, \tag{7}$$

where,

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,\, i' \in C_k} \sum_{j=1}^{T} \left(|x_{ij} - x_{i'j}| \right). \tag{8}$$

where,

$i, i' \in C_k$: Set of indices of observations in cluster C_k

$j \in T$: Set of time periods

$W(C_k)$: Within cluster variation of cluster C_k

$\left(\left|x_{ij} - x_{i'j}\right|\right)$: Absolute difference between mean value and observation i for time j

Equation (7) minimizes the total variation. Equation (8) calculates the within-cluster variation over the given time horizon. Since K-means clustering method requires us to define the number of clusters (K price scenarios) to be generated, we must decide the optimal number of price scenarios to be considered for SLP formulation.

3.2 Selection of optimal number of price scenarios

Elbow Method: This method looks at the percentage of variation as a function of the number of clusters [16]. As the number of clusters increases, cluster variation decreases. Therefore percentage change in cluster variation can be calculated between any two pairs of subsequent K values. Let us specify the threshold for change in cluster variation as 0.25%. Change in cluster variation with respect to the number of clusters is graphically expressed as follows:

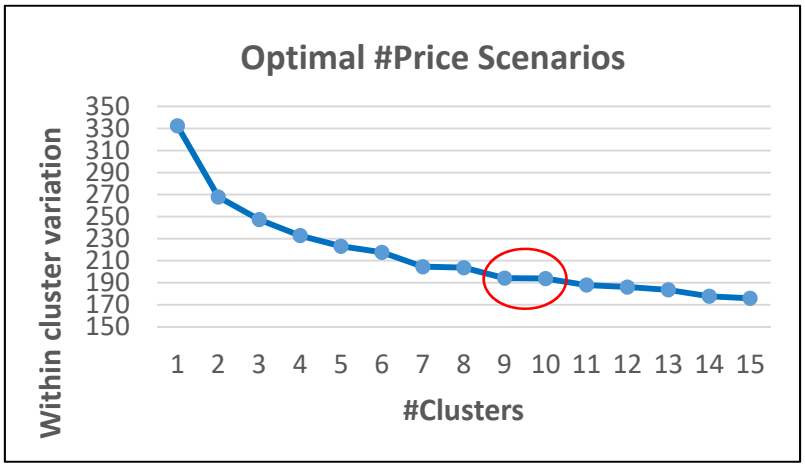


Figure 3: Elbow method for optimal price scenarios

The horizontal line between cluster no. 9 and 10 (red circled in the Figure 3) resembles an arm, therefore the "elbow" (the point of inflection on the curve) is a good indication that the underlying model fits best at that point.

Table 3: Changes in within-cluster variation

#Cluster:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Within cluster variation:	332.4	267.9	247.3	232.7	223.1	217.6	204.5	203.7	194.2	193.8	187.8	186.0	183.6	177.7	175.8
% change in variation:			7.68%	5.88%	4.15%	2.46%	6.01%	0.42%	4.67%	0.20%	3.05%	0.96%	1.34%	3.19%	1.05%

Table 3 shows the within-cluster variation values for k=1 to 15, and it is evident that the percentage variation is 0.20% between K value of 9 and 10, which is less than the threshold of 0.25%, therefore K=9 considered to be the optimal number of price scenarios, which can be represented graphically over 24 months' time horizon as follows:

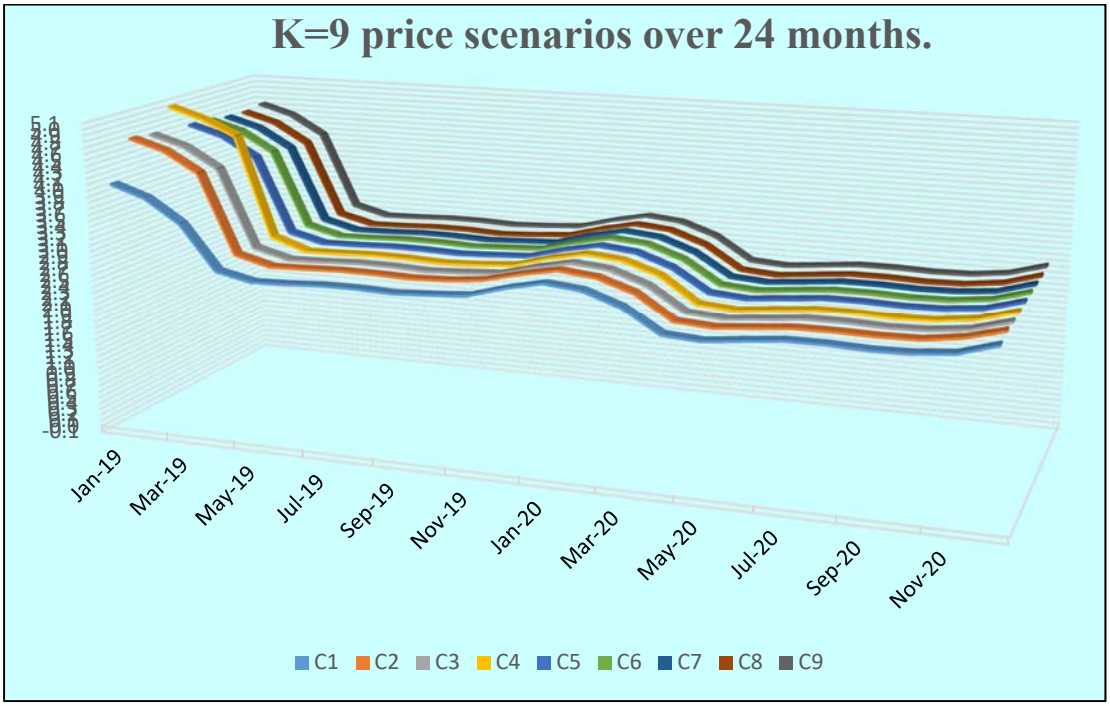


Figure 4: Result of K-means clustering with K=9 price Scenarios

Every price scenario (k = 1,…,9) is the mean of all the price vectors within that cluster, therefore the scenario probability (q_s) can be defined as the number of price vectors in cluster k over total available price scenarios, which can be mathematically expressed as:

$$q_s = \frac{Cluster\ Size}{Available\ price\ scenarios} = \frac{|C_k|}{|N|} \quad \forall k = 1, \dots, 9 \qquad (9)$$

Table 4: Calculation of scenario probabilities

	C1	C2	C3	C4	C5	C6	C7	C8	C9
Cluster size:	2	37	89	8	143	294	110	41	238
Available price scenarios	962	962	962	962	962	962	962	962	962
Scenario Probability:	0.002	0.038	0.093	0.008	0.149	0.306	0.114	0.043	0.247

3.3 SLP model with scenario probabilities

Objective function (1) can be reformulated by including the scenario probabilities (q_s) as follows:

$$maximize \sum_{s \in S} \sum_{t \in T} - p_{st} \,.\, x_{st} \,.\, q_s \qquad (10)$$

$$s.t.\ (2) – (6).$$

3.4 SLP model with scenario probabilities results

The model was implemented using the scenario probabilities (q_s) from table 3, operating characteristics from table 1 and the following results and policy decision profile were obtained:

Table 5: SLP model with K=9 price scenarios results

Expected Profit:	US$ 534,638
Computation Time in Seconds:	2.333

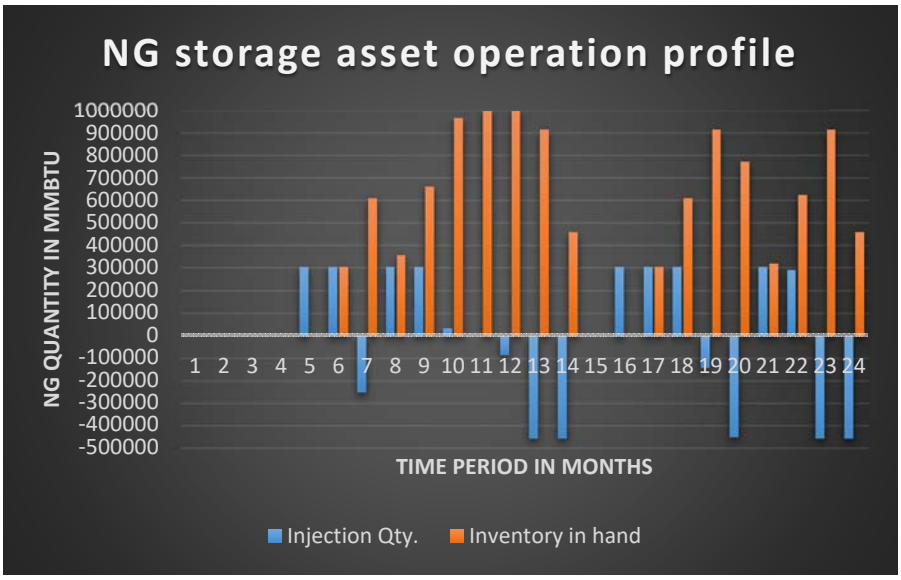


Figure 5: NG storage asset operation profile for SLP model K=9 price scenarios

We can see from Table 5 that the expected profit value is $534,638 which is just 0.1% smaller than the output of SLP model with 962 price scenarios. Since the number of scenarios significantly reduced from 962 to 9, and therefore the number of constraints, it just took couple of seconds for model to give the output. With such less computational time and deviation of just 0.1%, the SLP model with 9 price scenarios performed well and provides good approximate solution to the actual problem with 962 price scenarios. We can see from Figure 5 that the storage profile is very similar to the SLP model with 962 price scenarios, except no storage operation at time period 11, which resulted in total 18 storage operations, whereas the previous model gives 19 storage operations with some small injection quantity at time period 11.

Chapter 4: Stochastic Linear Programming model variants

4.1 Chance constrained model

The previous modeling methods were simple and provides the optimized solution for the given price scenarios, but in real practice, a user would want to know the reliability of the expected profit value resulted from the SLP formulation. Additionally, due to uncertainty in the price data, it is desirable to have some probabilistic guarantee of profit level. We utilized a chance constraint programming approach [3].

4.1.1 Model Formulation

The probabilistic constraint can be stated as:

$$Pr\left(\sum_{s \in S}\sum_{t \in T} - p_{st} \,.\, x_{st} \,.\, q_s \;\; \geq b\right) \geq \; \alpha, \tag{11}$$

where,

$\alpha =$ Confidence level

$b =$ Desired profit

Next, the constraint (11) can be converted to equivalent LP form by introducing the binary variable z_s , which takes value 1 when the condition is satisfied for scenario s, and 0 otherwise as follows:

$$\left(\sum_{t \in T} - p_{st}.x_{st} \right) - b \;\geq\; -M(1 - z_s) \qquad \forall s \in S \tag{12}$$

$s.t.$

$$\sum_{s \in S} q_s \,.\, z_s \;\geq \alpha \tag{13}$$

$$z_s \in [0, 1] \qquad \forall s \in S \tag{14}$$

and (2) – (6).

4.1.2 Chance constrained model results

The model was implemented by varying the desired profit level (b) and confidence level (α) along with the operating characteristics from table 1 and the results were plotted on the graph as follows:

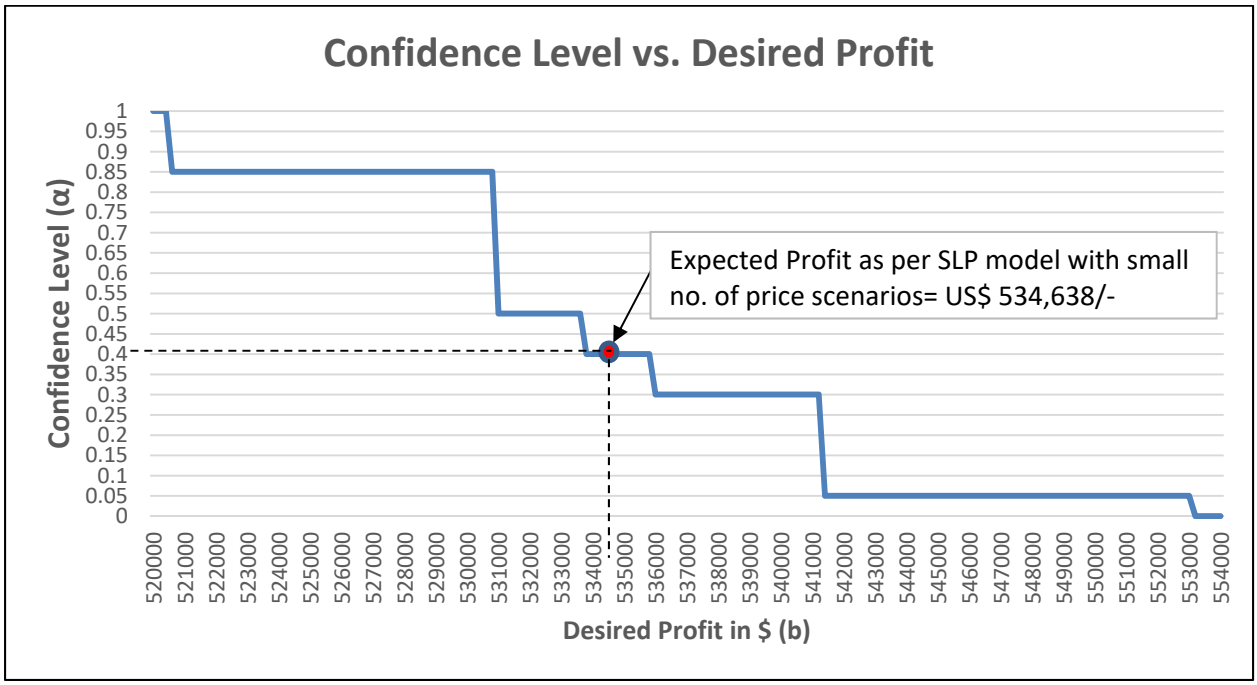


Figure 6: Maximum confidence level for varying desired profit values

As the desired profit level is increased, the maximum allowable confidence level decreases and the graph follows the step-wise profile due to the discrete price scenarios. Additionally, the graph shows that the confidence level for the expected profit value as per SLP model with K=9 price scenarios is 40%, which means that the probabilistic guarantee over expected profit value of US$ 534,638/- is 40% for the given K price scenarios.

4.2 Robust optimization model

While Stochastic optimization with chance constraint model in the previous section does account for confidence level over expected profit, it does not consider the situation where the

decision-maker is risk-averse. Sometimes, the case where user would want to operate the NG storage asset with least possible risk and still expect some profit at the end of the time horizon. Robust optimization method can be applied to create a risk-averse model.

4.2.1 Model Formulation

The objective function can be mathematically written as [12]:

$$\underset{x}{maximize}\ \ \underset{s}{minimize} \sum_{t \in T} - p_{st} \,.\, x_t \tag{15}$$

Here we can see that the (15) tries to maximize the profit by maximizing the x_t but at the same time considers the lowest price scenario s. It can be reformulated by including a positive continuous variable v as follows:

$$\underset{x,\ v}{maximize} \quad v \tag{16}$$

$s.t.$

$$v \leq - \sum_{t \in T} p_{st} \,.\, x_t \qquad \forall s \in S \tag{17}$$

$$and\ (2) - (6).$$

4.2.2 Robust optimization (RO) model results

The model was implemented using the operating characteristics from table 1 and the following results and policy decision profile were obtained:

Table 6: RO model results

	962 Price Scenarios	**9 Price Scenarios**
Expected Profit:	US$ 430,790	US$ 520,449
Computation Time in Seconds:	2.87	2.14

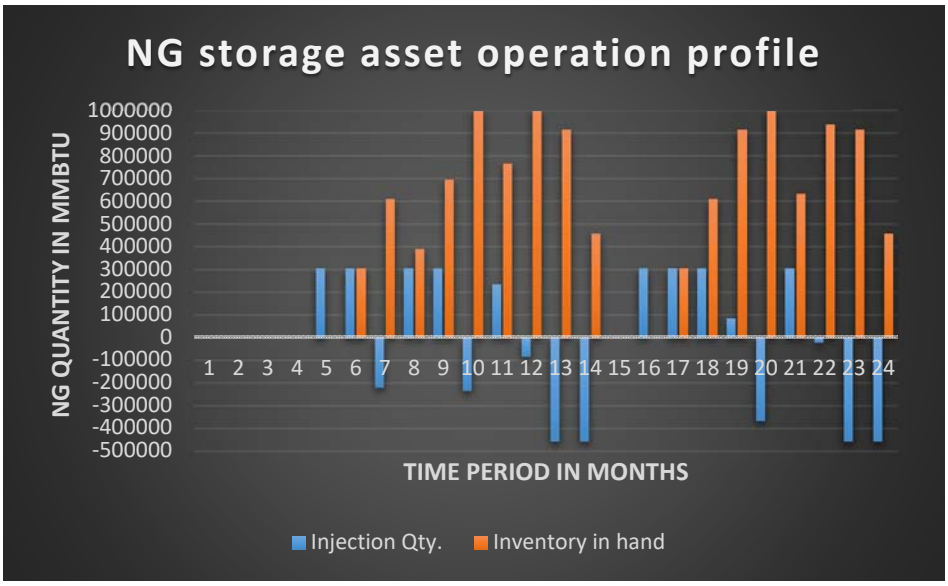


Figure 7: NG storage asset operation profile for RO model with 962 price scenarios

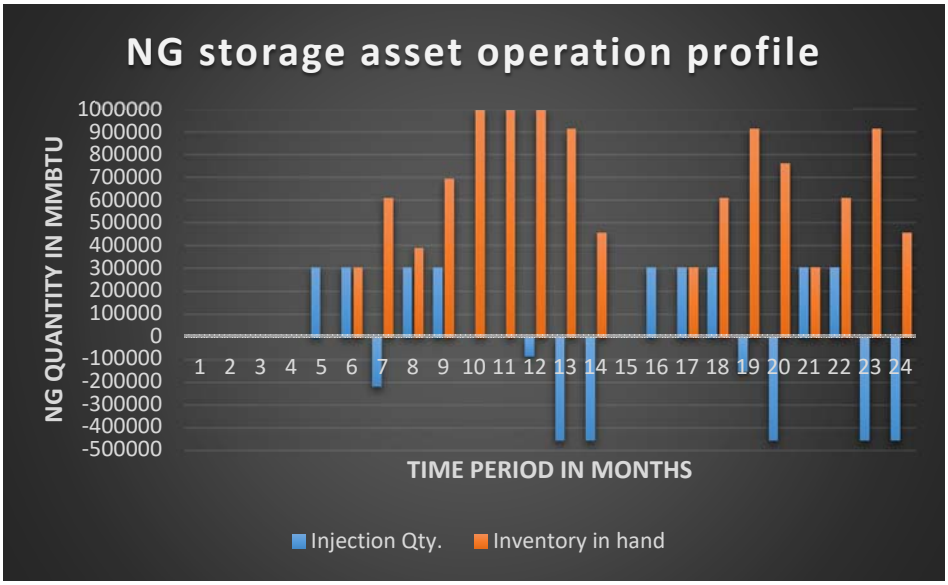


Figure 8: NG storage asset operation profile for RO model with K=9 price scenarios

The expected profit in the case of 9 price scenarios is around 20% more than that in the case of 962 price scenarios which is completely justifiable due to the large difference in numbers of price scenarios. If we consider every price scenario individually and solve the LP model, we get the minimum expected profit of $433,334 out of all 962 price scenarios, which is around 0.6% greater

than the output of RO model with 962 price scenarios. This means that RO model performed well with the expected profit of $430,790 which is $2544 smaller than the minimum of all 962 price scenarios. On the other hand, the expected profit value of RO model with 9 price scenarios exactly matches with the minimum of 9 price scenarios.

Figure 7 shows that the total number of storage operations in case of SLP-RO model with 962 price scenarios is 19 similar to the SLP model, however, the injection quantity varies during time period 10, 11, 19, 20 and 22 compared to the SLP model in order to account for worst-case. Similarly, Figure 8 shows that the total number of storage operations in case of SLP-RO model with 9 price scenarios is 17, whereas it was 18 for SLP model with 9 price scenarios. The injection quantity varies for time period 7, 8, 10, 19, 20 & 22 compared to the SLP model in order to account for worst-case.

4.3 Minimizing the maximum regret model

So far in this study, we tried to model the problem which was either too optimistic or too risk-averse (Robust optimization), although it can be the case where a decision-maker would want to take some reasonable amount of risk. The risk here is either underestimating or regretting the decision of being too optimistic. Regret can be phrased as the sense of loss felt by a decision-maker upon learning that an alternative action would have been preferable to the one actually selected [17]. Therefore, it is necessary to obtain the tradeoff between optimistic and robust optimization approach. The minimax regret solution proposed by [17] and [18] gives possibly optimal solution which minimizes the deviation from the necessary optimality.

4.3.1 Model Formulation

The objective function can be mathematically written as [17]:

$$minimize \quad r \tag{18}$$

$s.t.$

$$\sum_{t \in T} - p_{st} \,.\, x_t + r \;\geq\; \sum_{t \in T} - p_{st} \,.\, \bar{x}_{st} \qquad \forall s \in S \tag{19}$$

$$r \geq 0 \tag{20}$$

$$and \ (2) - (6).$$

Where, $\bar{x}_{st}$ = Injection/ withdrawal quantity of NG for time t for scenario s

Objective function (18) minimizes the maximum regret. Constraint (19) ensures the injection and withdrawal quantities x_t are selected which are less than that for all the price scenarios $\bar{x}_{st}$.

Parameter $\bar{x}_{st}$ is calculated using LP model as follows for all the price scenarios:

$$\bar{x}_{st} = \arg \max_{x_{st}} \; - \sum_{t \in T} p_{st} \,.\, x_{st} \qquad \forall s \in S \tag{21}$$

$$s.t. \ (2) - (6).$$

4.3.2 Minmax regret (MMR) model results

The model was implemented using the operating characteristics from table 1 and the following results and policy decision profile were obtained:

Table 7: MMR model results

	962 Price Scenarios	**9 Price Scenarios**
Expected Profit:	US$ 499,684	US$ 534,517
Computation Time in Seconds:	546.50	2.47

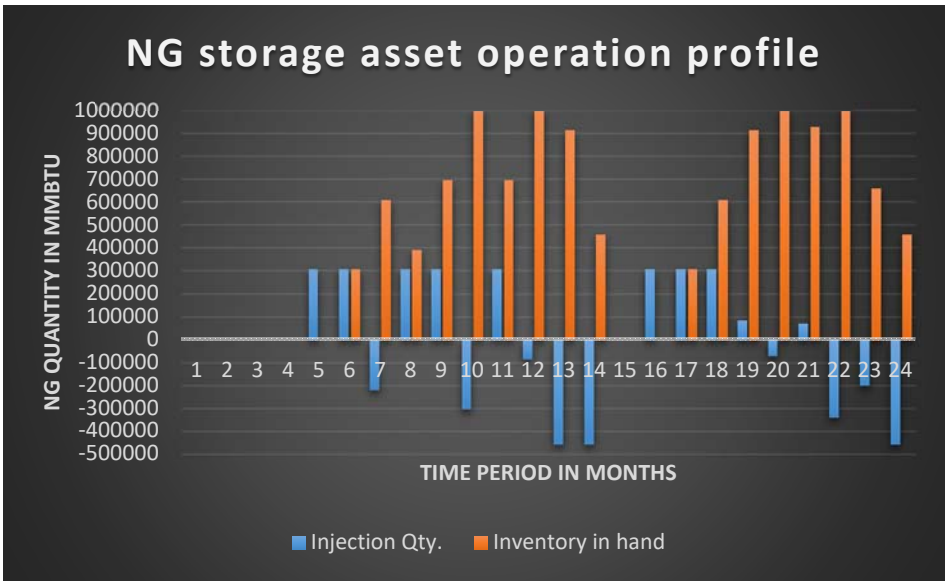


Figure 9: NG storage asset operation profile for MMR model with 962 price scenarios

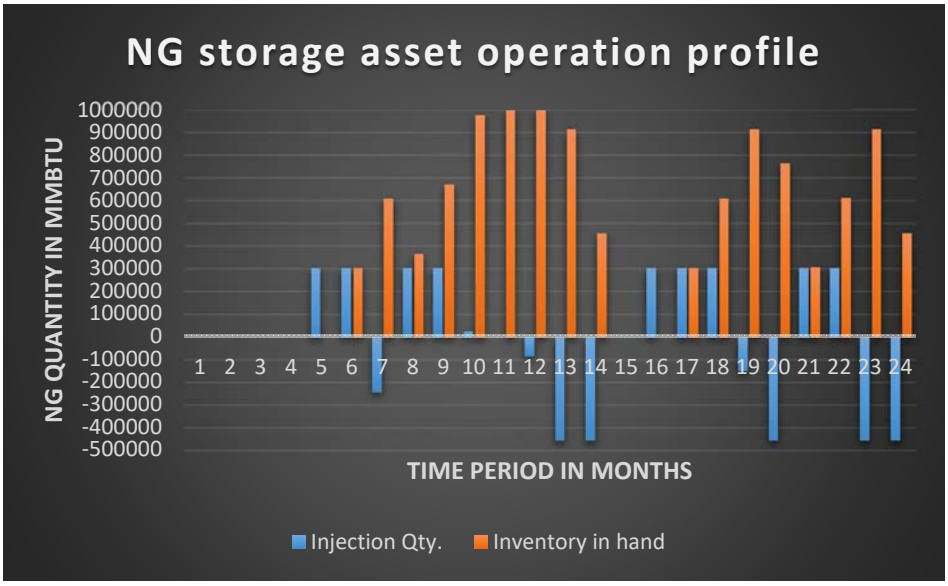


Figure 10: NG storage asset operation profile for MMR model with K=9 price scenarios

The expected profit in case of 9 price scenarios is around 7% more than that in case of 962 price scenarios which is completely justifiable due to large difference in numbers of price scenarios. If we compare the expected profit value of SLP-MMR model with SLP and SLP-RO model, for 962 price scenarios, the expected profit for SLP-MMR model is around 7% smaller than SLP model and around 13% greater than SLP-RO model. Whereas, for 9 price scenarios, the expected profit

for SLP-MMR model is around 0.02% smaller than SLP model and around 3% greater than SLP-RO model.

We can also see from figure 9 and 10 that the total number of storage operations in case of 962 scenarios is 19 whereas in case of 9 scenarios it is 18 and all the constraints (2 to 6, 20 and 21) are completely satisfied in both the cases.

Chapter 5: Deterministic Dynamic Programming

In contrast to linear programming models described in the previous chapters (2 and 3), the dynamic programming model can be solved optimally by breaking the problem into sub-problems and then recursively finding the optimal solutions to the sub-problem is said to have optimal substructure [11]. Based on Bellman's principle of optimality, any problem characterized as dynamic programming problem must have three features: Discreet set of stages, states and policy variables which is described in the next section [10].

5.1 Model Formulation

The continuous data used in linear programming model was converted to discrete sets as follows:

- $Discrete\ set\ of\ policy\ variables\colon\ x_i \in I\ (NG\ injection/withdrawal\ values)$
- $Discrete\ set\ of\ state\ variables\colon s_j \in J\ (NG\ storage\ values)$
- $Discrete\ set\ of\ stages\colon\ t \in T\ (Time\ periods)$

Unlike linear programming, there does not exist a standard mathematical formulation of the dynamic programming problem. Rather, dynamic programming is a general type of approach to problem-solving, and the particular equations used must be developed to fit each situation [11]. The computation starts from the last stage of the problem and follows the backward recursion, calculating the policy decisions at every stage for all the possible values of state variables until it reaches the first stage of the problem. Next, it starts taking the optimal decisions from the first stage in the forward direction until it reaches the final stage of the problem.

A simple three-stage example is formulated to conceptualize the problem as follows:

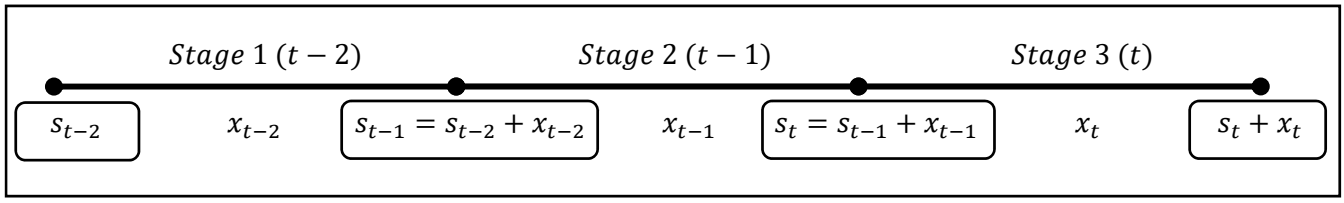
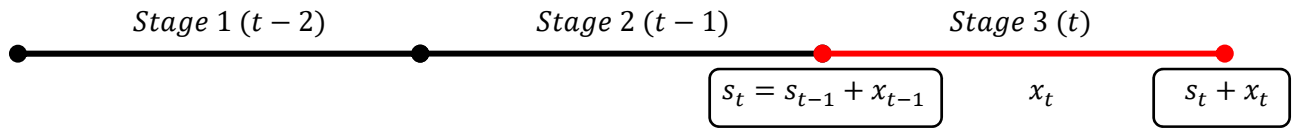


Figure 11: Three-stage dynamic programming example

- Discreet set of NG injection/withdrawal values: x_{t-2}, x_{t-1}, x_t
- Discreet set of NG storage values: $s_{t-2},\ s_{t-1},\ s_t$
- Discreet set of time periods: $t-2, t-1, t$

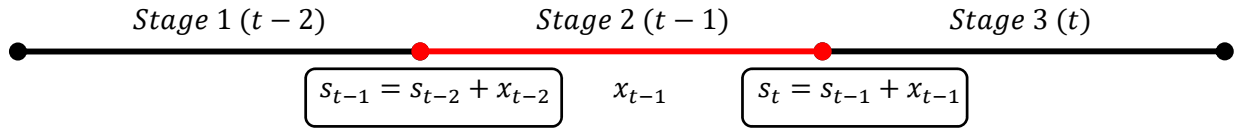
The recursive function equation for every stage is represented through Figure 12 below.

<u>*Stage* 3 (t):</u>



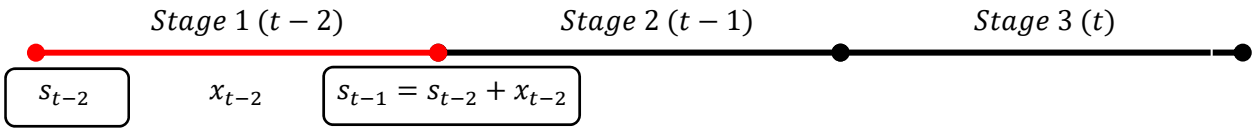
$$F^t(s_j, x_i) = max(-p_t \,.\, x_i) \qquad \forall\ (s_j + x_i) = s_T \tag{22}$$

<u>*Stage* 2 ($t-1$):</u>



$$F^{t-1}(s_j, x_i) = max\left[F^t(s_j, x_i) - p_{t-1} \,.\, x_i\right] \qquad \forall\ s_j \tag{23}$$

<u>*Stage* 1 ($t-2$):</u>



$$F^{t-2}(s_j, x_i) = max\left[F^{t-1}(s_j, x_i) - p_{t-2} \,.\, x_i\right] \qquad \forall\ s_j = s_0 \tag{24}$$

Figure 12: Recursive function equations

where,

p_{t-2}, p_{t-1}, p_t = Monthly NG prices for stage 1, 2 and 3, respectively

s_T = Storage level at the end of stage 3

s_T = Storage level at the beginning of stage 1

As shown in Figure 12, the equation (22) calculates the vectors of recursive functions for all the possible storage values s_j and equivalent policy variables x_i for stage 3 and similarly, equation (23) and (24) for stage 2 & stage 1, respectively. Also, stage 3 being the last stage of the problem, its recursive function (22) depends on the NG storage level at the beginning of stage 3 and the policy decisions taken, whereas the recursive functions of stage 1 and 2 (23) and (24) depends on the policy decisions taken in stage 2 and 3, respectively.

5.2 Deterministic dynamic programming model results

The above formulation was implemented using the following operating characteristics:

Table 8: Discretized Operating Characteristics

$x_i \in I$	$-500000, \ -499000, \ldots, 305000$
$s_j \in J$	$0, 1000, 2000, \ldots, 1000000$
$t \in T$	$1, 2, 3, \ldots, 24$
S_0	0
S_T	0

Using the provided data, the following results and policy decision profile were obtained:

Table 9: DDP model results

Expected Profit:	US$ 555,809
Computation Time in Seconds:	18553.9 sec. ~ 5hrs.

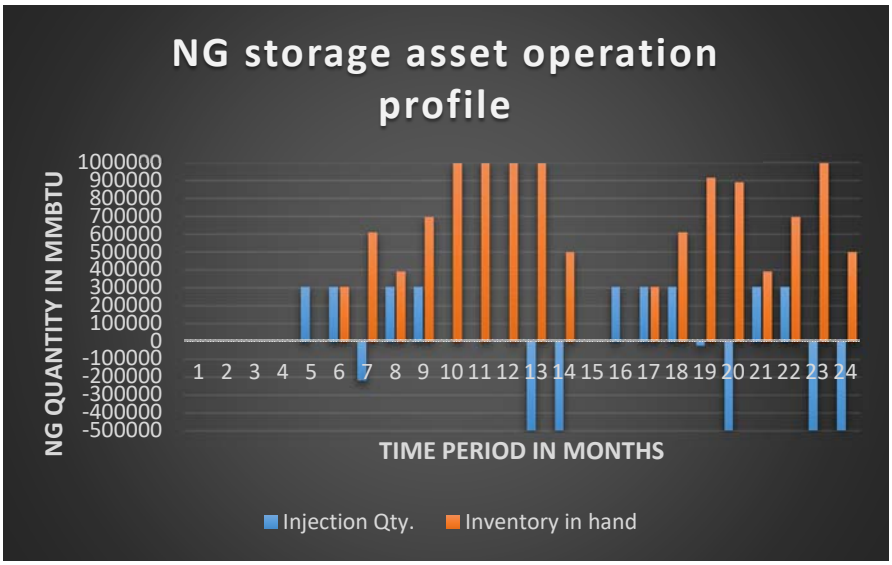


Figure 13: NG storage asset operation profile for DDP model

We can see from Figure 13 that the storage, injection, and withdrawal values are all multiples of 1000 because the discreet values of sets in the operating characteristics (Table 8) are considered in the step size of 1000. If we change the step size of discreet data from 1000 to 10,000, we get the same output but computational time goes down substantially from ~ 5hours to just over 3 seconds. If we approximate the maximum NG withdrawal quantity for a single time period from 457,500 to 500,000 and solve the Pauls' MILP model, we get the exact same output as DDP.

Chapter 6: Model Summary

Table 10: Model Summary

Model variant / Time period	962 Price Scenarios			K=9 Price Scenarios			DDP
	SLP	MMR-SLP	RO-SLP	SLP	MMR-SLP	RO-SLP	
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	305000	305000	305000	305000	305000	305000	305000
6	305000	305000	305000	305000	305000	305000	305000
7	-220000	-220000	-220000	-252978	-244448	-220000	-220000
8	305000	305000	305000	303347	305000	305000	305000
9	305000	305000	305000	305000	305000	305000	305000
10	0	-305000	-234748	34631	24448	0	0
11	0	305000	234748	0	0	0	0
12	-85000	-85000	-85000	-85000	-85000	-85000	0
13	-457500	-457500	-457500	-457500	-457500	-457500	-500000
14	-457500	-457500	-457500	-457500	-457500	-457500	-500000
15	0	0	0	0	0	0	0
16	305000	305000	305000	305000	305000	305000	305000
17	305000	305000	305000	305000	305000	305000	305000
18	305000	305000	305000	305000	305000	305000	305000
19	-152500	85000	85000	-143365	-151746	-152500	-25000
20	-457500	-72369.385	-367723	-451635	-457500	-457500	-500000
21	305000	72369.385	305000	305000	305000	305000	305000
22	305000	-340952.92	-22277	290000	304246	305000	305000
23	-457500	-201547.08	-457500	-457500	-457500	-457500	-500000
24	-457500	-457500	-457500	-457500	-457500	-457500	-500000
Expected Profit:	**$535,198.00**	**$499,683.76**	**$430,790.39**	**$534,637.88**	**$534,516.88**	**$520,448.72**	**$555,809.19**
Computational time:	**11.89 sec**	**546.50 sec**	**2.87 sec**	**2.33 sec**	**2.471 sec**	**2.14 sec**	**30.4 sec.**

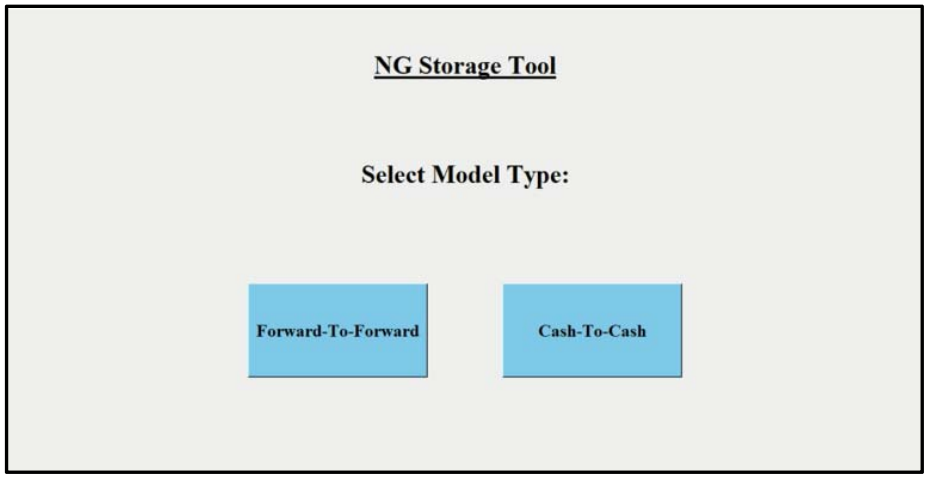
Table 10 summarizes the output of all the models. The negative figures indicates the withdrawal quantities of NG. As expected, the SLP model gives the best solution for both, 962 and 9 price scenarios compare to MMR-SLP and RO-SLP models. The RO-SLP model being the most conservative (risk-averse) modeling approach gives the lowest expected profit values, whereas, the MMR-SLP being the risk-neutral modeling approach gives the moderate expected profit values.

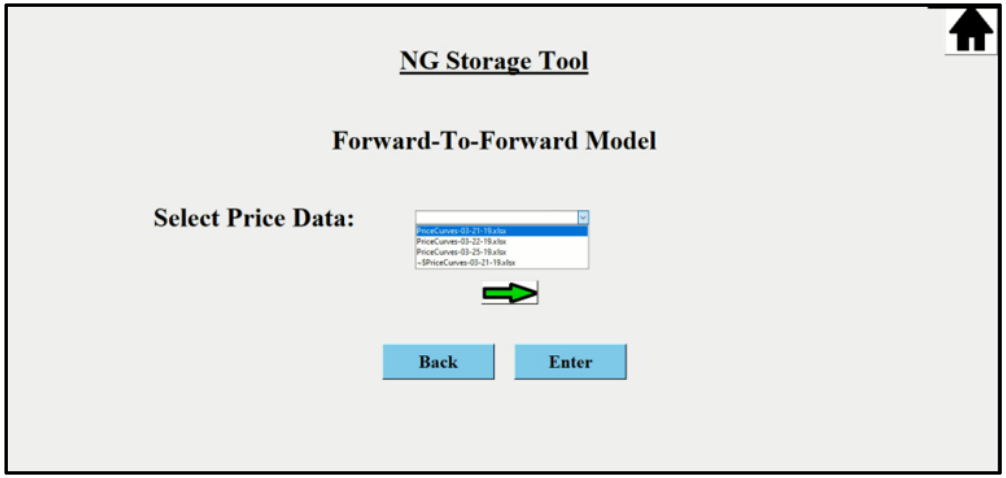
The difference in output between SLP model with 962 price scenarios and 9 price scenarios is not significant because the NG prices are concentrated within a small range (2.2722 to 5.4874).

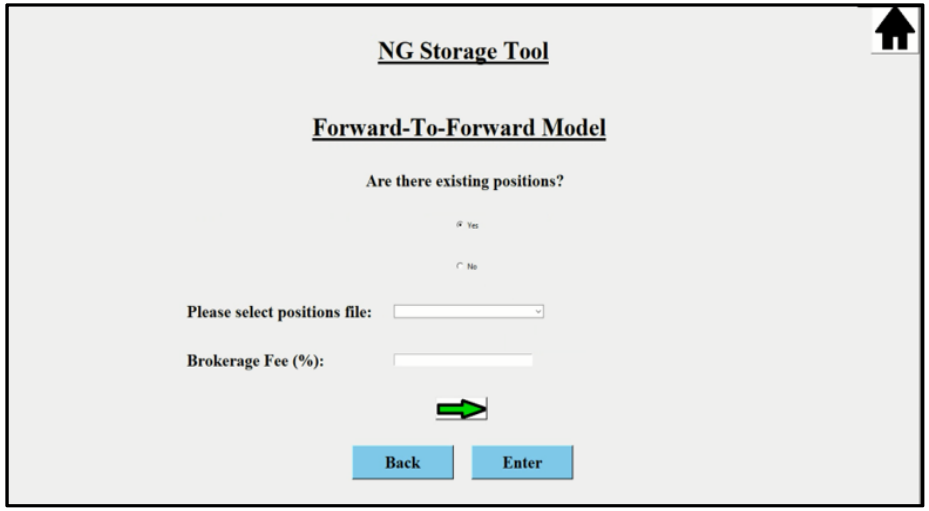
In case of DDP, the policy variables (the discreet range of maximum withdrawal and injection quantities) were in a step size of 1000 (-500000, -499000, …., 305000), whereas, SLP models considered the continuous variable (x) ranging from -457500 to +305000. Therefore, the expected profit value derived from DDP model can not be directly compared with that of SLP mode

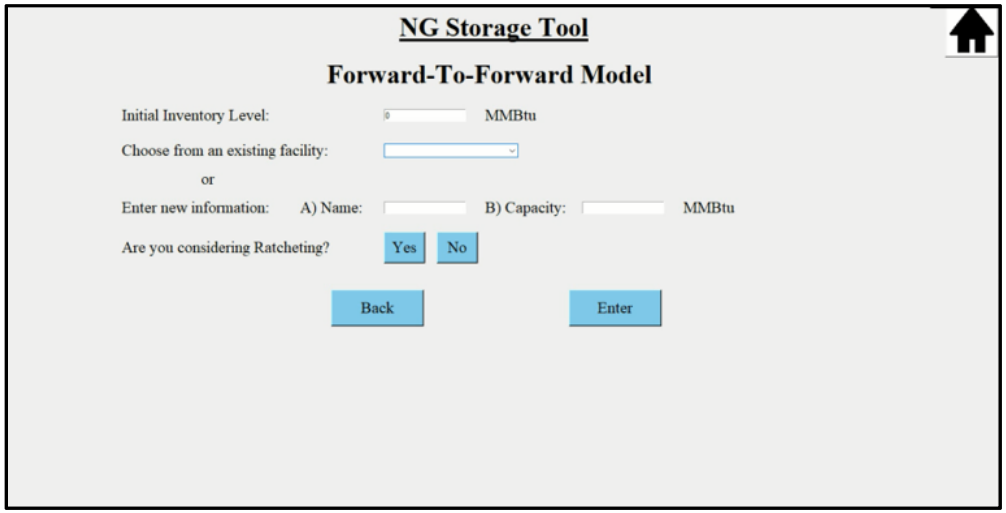
Chapter 7: User Interface for LP models

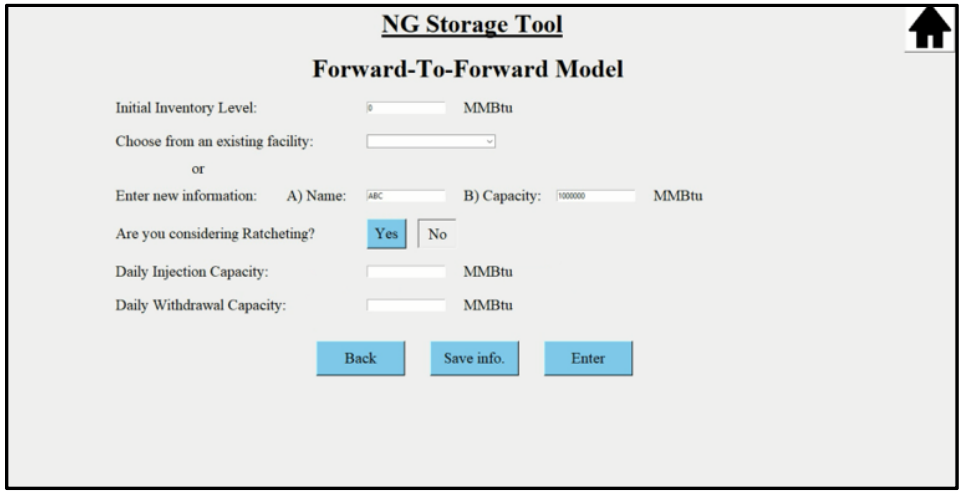
The Graphic User Interface tool was developed using Python module-Tkinter for the industry client to interact with the LP models developed by Paul. Few snapshots of the actual tool are as shown below:

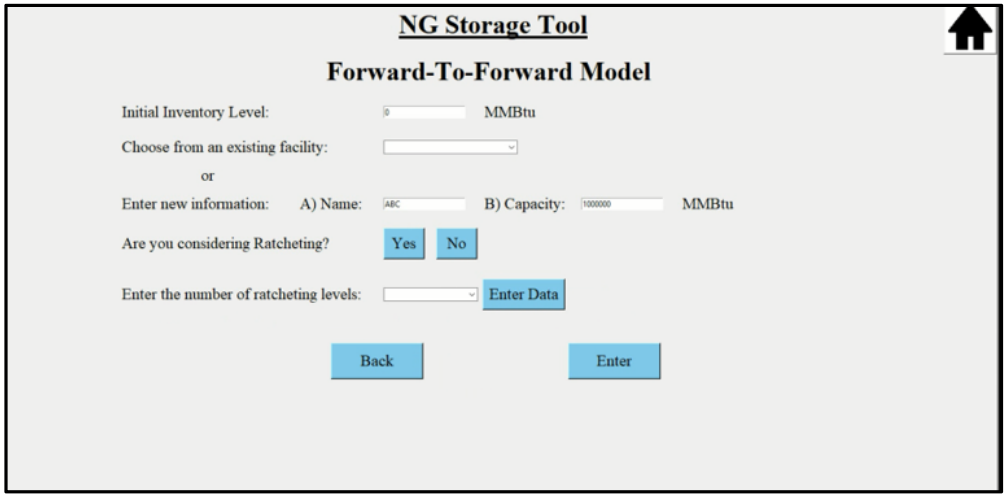


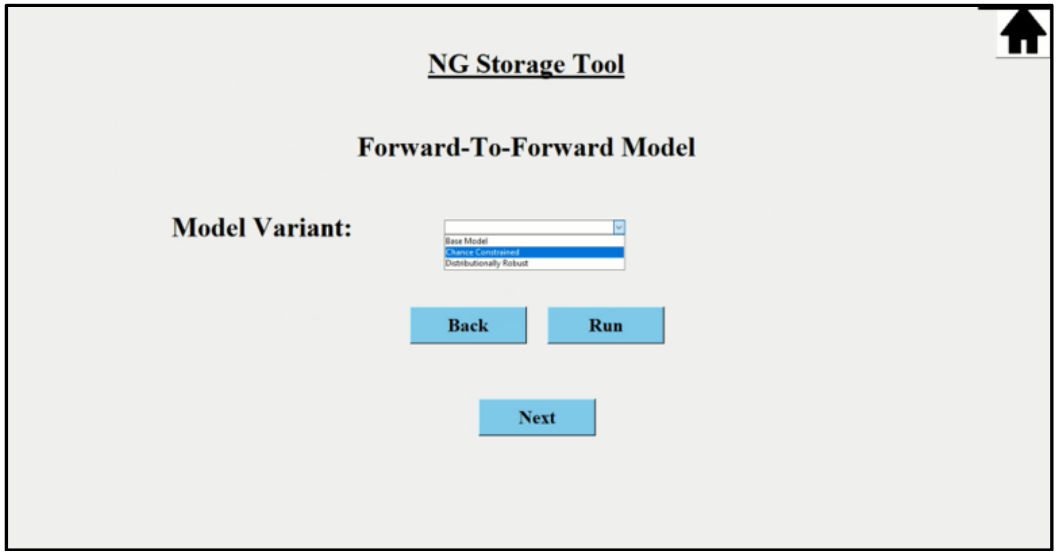







NG Storage Tool
Forward-To-Forward Model
Initial Inventory Level:
0
MMBtu
Choose from an existing facility:
or
Enter new information:
A) Name:
ABC
B) Capacity:
1000000
MMBtu
Are you considering Ratcheting?
Yes
No
Daily Injection Capacity:
MMBtu
Daily Withdrawal Capacity:
MMBtu
Back
Save info.
Enter


NG Storage Tool
Forward-To-Forward Model
Initial Inventory Level:
0
MMBtu
Choose from an existing facility:
or
Enter new information:
A) Name:
ABC
B) Capacity:
1000000
MMBtu
Are you considering Ratcheting?
Yes
No
Enter the number of ratcheting levels:
Enter Data
Back
Enter


NG Storage Tool
Forward-To-Forward Model
Model Variant:
Base Model
Chance Constrained
Distributionally Robust
Back
Run
Next

NG Storage Tool

Forward-To-Forward Model: Distributionally Robust

Select Risk-Adversity:

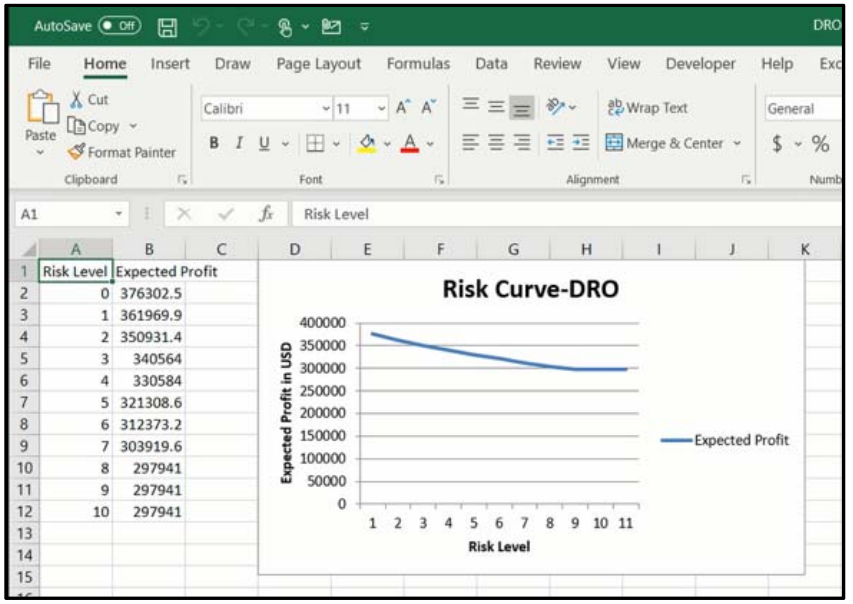
0 1 2 3 4 5 6 7 8 9 10

Run

or

Generate Risk Curve: Run

Choose Another Model Variant | End Session



NG Storage Tool

Forward-To-Forward Model: Chance Constrained

Desired Profit Level:

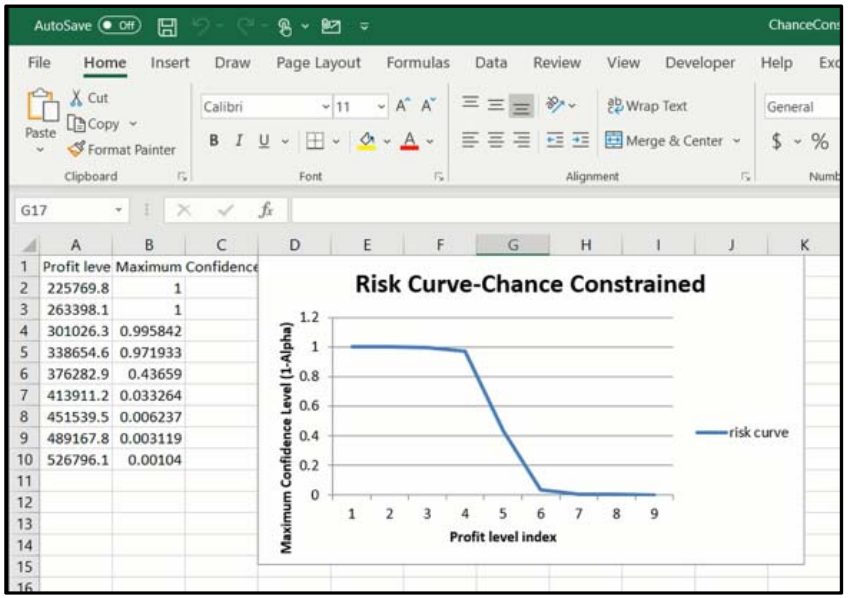
and Run

Desired % Confidence in Meeting Profit Level:

or

Generate Risk Curve: Run

Choose Another Model Variant | End Session



Chapter 8: Conclusion

Throughout the project, several optimization models were developed to aid in NG storage decisions. The optimization models were created based on the operating characteristics and constraints of NG storage asset provided by the industry client. Clustering algorithm was implemented over a large price data to obtain the minimum number of price scenarios and elbow method to get the optimal K value (optimal price scenarios). SLP model, including the scenario probabilities, were implemented to get the best solution. In order to back the output of SLP model with some level of confidence, SLP model was then further extended by incorporation of chance constraint, which resulted in probabilistic guarantee of 40% over the expected profit value.

NG storage is one of the vital element of the complete supply chain network and approximately 50% and 62% of the value of storage can be attributed to natural gas volatility (stochastic variability)[19]. We introduced some model variants to account for the price uncertainty. Considering the worst-case scenario, Robust Optimization model with the lowest possible expected output was considered, which resulted in most conservative NG storage asset operation profile. On the other hand, in order to deal with the situation where decision-maker would be ready to take some calculative risk, the MiniMax regret model (tradeoff between optimistic and robust optimization approach) was considered. As expected, the output of MiniMax Regret model was moderate.

Additionally, Deterministic Dynamic Programming (DDP) model was implemented to aid the problem. Although the output derived was satisfactory, the computation time taken by DDP model was very high (~5hours) compared to the SLP model variants (few seconds). Also, some

deviation in NG storage asset operation profile was observed in comparison with SLP model variants due to discreet set of storage and policy variables data.

At the end, the snapshots of NG storage operation tool (GUI package) developed for the industry client to interact with the LP models (developed by Paul Jobinpicard in his thesis) was presented.

References

[1] NG stats report issued by Canadian Gas Association on January'2018

[2] Avery, W., Brown, G. G., Rosenkranz, J. A., & Wood, R. K. (1992). Optimization of Purchase, Storage and Transmission Contracts for Natural Gas Utilities. Operations Research, 40(3), 446-462. doi:10.1287/ opre.40.3.446

[3] Ahmed, Shabbir & Shapiro, Alexander (2008). Solving Chance-Constrained Stochastic Programs via Sampling and Integer Programming. Tutorials in Operations Research.

[4] Thompson, M., Davison, M., & Rasmussen, H. (2009). Natural gas storage valuation and optimization: A real options application. Naval Research Logistics, 56(3), 226-238. doi:10.1002/nav.20327

[5] Sturm, F. J. (1997). Trading natural gas cash futures options and swaps. Tulsa (Okla.): PennWell Books.

[6] Guldmann, J., & Fahui, W. (1999). Optimizing the natural gas supply mix of local distribution utilities. European Journal of Operations Research 112 (1999) 598-612

[7] Reuven Levary, R., & Dean, B., (1980). A Natural Gas Flow Model under Uncertainty in Demand. Operations Research, Vol. 28, No. 6 (Nov. - Dec., 1980), pp. 1360-1374

[8] BOPP A.E., KANNAN, V.R., PALOCSAY, S.W., & STEVENS, S.P., (1996). An Optimization Model for Planning Natural Gas Purchases, Transportation, Storage, and Deliverability. Omega, Vol. 24 No. 5, pp. 511-522, 1996

[9] Jobinpicard, P., (2019). Emera Energy Storage Optimization Project (Thesis Report)

[10] Nandalal, K., & Bogardi, J. Dynamic programming based operation of reservoirs. Cambridge University Press

[11] Hillier, F., & Lieberman, G. (2007). Introduction to Operations Research (Tenth Edition): McGraw-Hill Education

[12] Ben-Tal, A., Ghaoui, L. E., & Nemirovski, A. (2009). Robust optimization. Princeton: Princeton University Press

[13] Jong, C., (2015). Gas Storage Valuation and Optimization. Journal of Natural Gas Science and Engineering 24 (2015) 365e37

[14] Fancher, R., Wilson, J., & Mueller, H., (1986). Fuel Contracting Under Uncertainty. IEEE Transactions on Power Systems, Vol. PWRS-1, No. 1

[15] James, G., Witten, D., Hastie, T., & Tibshirani, R., (2013). An Introduction to Statistical Learning. Springer New York Heidelberg Dordrecht London

[16] Elbow Method- https://www.scikit-yb.org/en/latest/api/cluster/elbow.html]

[17] Mausser, H., & Laguna, M., (1998). A New Mixed Integer Formulation for the Maximum Regret Problem. International Transactions in Operational Research Vol. 5, No. 5.

[18] Inuiguchi, M., Higashitani, H., & Tanino, T., (1999). On Computation Methods of the Minimax Regret Solution for Linear Programming Problems with Uncertain Objective Function Coefficients. Graduate School of Engineering, Osaka University, Japan

[19] Lai, G., Wang, M., Kekre, S., Scheller-Wolf, A., & Secomandi, N., (2011). Valuation of Storage at a Liquefied Natural Gas Terminal. Operations Research, Vol. 59, No. 3, pp. 602-616

Appendix

1. SLP Model:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook
import numpy as np
import pandas as pd

wb=load_workbook(filename= 'Price Curves 12-5-18.xlsx')
ws1=wb['Sheet2']

M=1000000
Cap=1000000
cin=305000
cout=-457500
d=1
Horizon=24
N=962

T=list(range(Horizon))
S=list(range(N))

p=[[0 for t in range(Horizon)] for s in range(N)]
q=[0 for s in range(N)]

for t in range(Horizon):
    for s in range(N):
        p[s][t]=ws1.cell(row=t+2, column=s+2).value

for s in range(N):
    q[s]=(1/N)

prob= LpProblem("EL-NR", LpMaximize)
x= LpVariable.matrix("x", (S,T), None, None, LpContinuous)
I= LpVariable.matrix("I", (S,T), 0, None, LpContinuous)

prob+= lpSum([lpSum([-d*x[s][t]*p[s][t]*q[s] for t in T]) for s in S])
for s in S:
    for t in T:
        prob+=cout<=x[s][t]
        prob+=x[s][t]<=cin
        prob+=I[s][t]<=Cap
        if t>0:
```

```
            prob+=I[s][t]==I[s][t-1]+x[s][t-1]
    prob+= I[s][Horizon-1]+x[s][Horizon-1]==I[s][0]
    prob+= I[s][0]==0
prob.solve()

ylist=list()
for s in S:
    for v in prob.variables():
        if v.name[:3]=='x_'+str(s):
            print(v.name, "=", v.varValue)
            ylist.append(v.varValue)

print(prob.objective.value())

df= pd.DataFrame(ylist[x:x+24] for x in range(0, len(ylist), 24)).transpose()
writer = pd.ExcelWriter('Xst of SLP.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', startcol=0, startrow=0, header=False,
index=False)
writer.save()

print("--- %s seconds ---" % (time.time() - start_time))
```

2. K means clustering algorithm:

```
#matrix math
import numpy as np
#graphing
import matplotlib.pyplot as plt
#graphing animation
import matplotlib.animation as animation
from scipy.spatial.distance import squareform, pdist
import scipy
import pandas as pd
from openpyxl import load_workbook
from collections import defaultdict
import math

#load data textfile
def load_dataset(name):
    return np.loadtxt(name)

#manhattan distance between 2 data points. For as many data points as necessary.
def euclidian(a, b):
    return np.linalg.norm(a-b,1)  #manhattan

def kmeans(k, epsilon=0, distance='euclidian'):
    #list to store past centroid
```

```
history_centroids = []
#set the distance calculation type
if distance == 'euclidian':
    dist_method = euclidian
#set the dataset
dataset = load_dataset('data.txt')
# dataset = dataset[:, 0:dataset.shape[1] - 1]
#get the number of rows (instances) and columns (features) from the dataset
num_instances, num_features = dataset.shape
#define k centroids (how many clusters do we want to find?) chosen randomly
prototypes = dataset[np.random.randint(0, num_instances - 1, size=k)]
#set these to our list of past centroid (to show progress over time)
history_centroids.append(prototypes)
#to keep track of centroid at every iteration
prototypes_old = np.zeros(prototypes.shape)
#to store clusters
belongs_to = np.zeros((num_instances, 1))
norm = dist_method(prototypes, prototypes_old)
iteration = 0
while norm > epsilon:
    iteration += 1
    norm = dist_method(prototypes, prototypes_old)
    prototypes_old = prototypes
    #for each instance in the dataset
    for index_instance, instance in enumerate(dataset):
        #define a distance vector of size k
        dist_vec = np.zeros((k, 1))
        #for each centroid
        for index_prototype, prototype in enumerate(prototypes):
            #compute the distance between x(data points) and centroid
            dist_vec[index_prototype] = dist_method(prototype, instance)
        #find the smallest distance, assign that distance to a cluster
        belongs_to[index_instance, 0] = np.argmin(dist_vec)

    tmp_prototypes = np.zeros((k, num_features))

    #for each cluster, k of them
    for index in range(len(prototypes)):
        #get all the points assigned to a cluster
        instances_close = [i for i in range(len(belongs_to)) if belongs_to[i] == index]
        #find the mean of those points, this is our new centroid
        prototype = np.mean(dataset[instances_close], axis=0)
        # prototype = dataset[np.random.randint(0, num_instances, size=1)[0]]
        #add out new centroid to our new temporary list
        tmp_prototypes[index, :] = prototype

    #set the new list to the current list
```

```
        prototypes = tmp_prototypes

        #add our calculated centroids to our history of plotting
        history_centroids.append(tmp_prototypes)

    #return calculated centroids, history of them all, and assignments for
    return prototypes, history_centroids, belongs_to

#main file
def execute():
    #load dataset
    clusters=20
    for z in list(range(1,clusters+1)):

        dataset = load_dataset('durudataset.txt')
        centroids, history_centroids, belongs_to = kmeans(z)

        wb=load_workbook(filename= 'pricedata.xlsx')
        ws=wb['Sheet1']
        p=[[0 for j in range(24)] for i in range(962)]
        for i in range(962):
            for j in range(24):
                p[i][j]=ws.cell(row=i+2, column=j+2).value

        B= list(belongs_to.flatten())
        C=[]
        for b in B:
            C.append(int(b))

        d = defaultdict(list)
        for key, value in zip(C, p):
            d[key].append(value)
        cluster_numbers=list()
        withinness=list()
        cluster_size=list()
        for m in list(range(z)):
            xxxx=list()
            yyyy=list()
            for k in range(len(dict(d)[m])):
                H=[abs(j-i) for i, j in zip(history_centroids[-1][m], dict(d)[m][k])]
                J=sum(H)
                xxxx.append(H)
                yyyy.append(J)
            cluster_size.append(len(dict(d)[m]))
            withinness.append(sum(yyyy))

        df1= pd.DataFrame(data=withinness)
```

```
        df2= pd.DataFrame(data=[sum(withinness)])
        df3= pd.DataFrame(data=history_centroids[-1].flatten())
        df4= pd.DataFrame(data=cluster_size)
        df5= pd.DataFrame(data=belongs_to.flatten())


        writer = pd.ExcelWriter('C:\Sixth Term @Dal\k_means_clustering-
master\kmeans_manhattan clusters\\'+str('Cluster')+str(z)+'.xlsx', engine='xlsxwriter')
        df1.to_excel(writer, sheet_name='cluster_withinnesss', startcol=0, startrow=0,
header=False, index=False)
        df2.to_excel(writer, sheet_name='Total_withinness', startcol=0, startrow=0,
header=False, index=False)
        df3.to_excel(writer, sheet_name='centroids', startcol=0, startrow=0, header=False,
index=False)
        df4.to_excel(writer, sheet_name='cluster_size', startcol=0, startrow=0, header=False,
index=False)
        df5.to_excel(writer, sheet_name='belongs_to', startcol=0, startrow=0, header=False,
index=False)
        writer.save()


#do everything
execute()
```

3. SLP model with K price scenarios:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook

wb=load_workbook(filename= 'Compiled output.xlsx')
ws=wb['kmeans']
ws1=wb['price data']

M=1000000
Cap=1000000
cin=305000
cout=-457500
d=1
Horizon=24
N=9

T=list(range(Horizon))
S=list(range(N))

p=[[0 for t in range(Horizon)] for s in range(N)]
```

```
q=[0  for s in range(N)]

for t in range(Horizon):
    for s in range(N):
        p[s][t]=ws.cell(row=t+6, column=s+38).value

for s in range(N):
    q[s]=ws.cell(row=31, column=s+38).value

prob= LpProblem("EL-NR", LpMaximize)
x= LpVariable.matrix("x", (S,T), None, None, LpContinuous)
I= LpVariable.matrix("I", (S,T), 0, None, LpContinuous)

prob+= lpSum([lpSum([-d*x[s][t]*p[s][t]*q[s] for t in T]) for s in S])

for s in S:
    for t in T:
        prob+=cout<=x[s][t]
        prob+=x[s][t]<=cin
        prob+=I[s][t]<=Cap
        if t>0:
            prob+=I[s][t]==I[s][t-1]+x[s][t-1]
    prob+= I[s][Horizon-1]+x[s][Horizon-1]==I[s][0]
    prob+= I[s][0]==0
prob.solve()

print(prob.objective.value())
for v in prob.variables():
    if v.name[:1]=='x':
        print(v.name, "=", v.varValue)

print("--- %s seconds ---" % (time.time() - start_time))
```

4. SLP model with chance constraint optimization:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook
import pandas as pd

wb=load_workbook(filename= 'Compiled output.xlsx')
ws=wb['kmeans']

M=1000000
Cap=1000000
```

```
cin=305000
cout=-457500
d=1
Horizon=24
T=list(range(24))
S=list(range(9))

p=[[0 for t in range(24)] for s in range(9)]
q=[0  for s in range(9)]

for t in range(24):
    for s in range(9):
        p[s][t]=ws.cell(row=t+6, column=s+38).value

for s in range(9):
    q[s]=ws.cell(row=31, column=s+38).value

OFlist2=list()
b=520000
while b <= 554000:
    OFlist1=list()
    alpha=0
    while alpha <=1:
        b=round(b,2)
        alpha=round(alpha,2)
        prob= LpProblem("EL-NR", LpMaximize)
        zz = LpVariable.matrix("zz", (S), 0,1, LpInteger)
        x= LpVariable.matrix("x", (S,T), None, None, LpContinuous)
        I= LpVariable.matrix("I", (S,T), 0, None, LpContinuous)

        prob+= lpSum([lpSum([-d*x[s][t]*p[s][t]*q[s] for t in T]) for s in S])

        for s in S:
            for t in T:
                prob+=cout<=x[s][t]
                prob+=x[s][t]<=cin
                prob+=I[s][t]<=Cap
                if t>0:
                    prob+=I[s][t]==I[s][t-1]+x[s][t-1]
            prob+= I[s][Horizon-1]+x[s][Horizon-1]==I[s][0]
            prob+= I[s][0]==0
        for s in S:
            prob+=lpSum([-d*x[s][t]*p[s][t] for t in T])-b>= -M*(1-zz[s])
        prob+=lpSum([q[s]*zz[s] for s in S])>=alpha
        prob.solve(GUROBI())
        OFlist1.append(prob.objective.value())
```

```
        alpha+=0.05
    b+=100
    OFlist2.append(OFlist1)

df=pd.DataFrame(OFlist2).transpose()
writer = pd.ExcelWriter('Chance1.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', startcol=0, startrow=0, header=False, index=True)
writer.save()

print("--- %s seconds ---" % (time.time() - start_time))
```

5. SLP with RO model:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook

wb=load_workbook(filename= 'Compiled output.xlsx')
ws=wb['kmeans']         # For 9 scenarios
ws1=wb['price data']    # For 962 scenarios

M=1000000
Cap=1000000
cin=305000
cout=-457500
d=1
Horizon=24
N=962

T=list(range(Horizon))
S=list(range(N))

p=[[0 for t in range(Horizon)] for s in range(N)]
q=[0  for s in range(N)]

for t in range(Horizon):
    for s in range(N):
#        p[s][t]=ws.cell(row=t+6, column=s+38).value    # For 9 scenarios
        p[s][t]=ws1.cell(row=t+2, column=s+2).value     # For 962 scenarios

prob= LpProblem("EL-NR", LpMaximize)

x= LpVariable.matrix("x", (T), None, None, LpContinuous)
I= LpVariable.matrix("I", (T), 0, None, LpContinuous)
v= LpVariable('v',  0, None, LpContinuous)
```

```
prob+= v

for s in S:
    prob+= v<=lpSum([-d*x[t]*p[s][t] for t in T])

for t in T:
    prob+=cout<=x[t]
    prob+=x[t]<=cin
    prob+=I[t]<=Cap
    if t>0:
        prob+=I[t]==I[t-1]+x[t-1]

prob+= I[Horizon-1]+x[Horizon-1]==I[0]
prob+= I[0]==0
prob.solve()

print(prob.objective.value())

for v in prob.variables():
    if v.name[:1]=="x":
        print(v.name, "=", v.varValue)

print("--- %s seconds ---" % (time.time() - start_time))
```

6. SLP with MMR model:

Calculating the parameter $\bar{x}_{st}$:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook
import pandas as pd

#For 9 Clusters#
#wb=load_workbook(filename= 'Compiled output.xlsx')
#ws=wb['kmeans']

#For 962 clusters#
wb1=load_workbook(filename= 'Price Curves 12-5-18.xlsx')
ws1=wb1['Sheet2']

M=10000000
Cap=1000000
cin=305000
cout=-457500
d=1
```

```
Horizon=24
N=962 #For 962 Clusters
#N=9 #For 9 Clusters

T=list(range(Horizon))
S=list(range(N))

p=[0 for t in range(Horizon)]
ylist=list()
for s in S:
    for t in range(Horizon):
#        p[t]=ws.cell(row=t+6, column=38+s).value    #For 9 Clusters
        p[t]=ws1.cell(row=t+2, column=2+s).value     #For 962 Clusters
    prob= LpProblem("EL-NR", LpMaximize)
    x= LpVariable.matrix("x", (T), None, None, LpContinuous)
    I= LpVariable.matrix("I", (T), 0, None, LpContinuous)

    prob+= (lpSum([-p[t]*x[t] for t in T]))
    for t in T:
        prob+=cout<=x[t]
        prob+=x[t]<=cin
        prob+=I[t]<=Cap
        if t>0:
            prob+=I[t]==I[t-1]+x[t-1]

        prob+= I[Horizon-1]+x[Horizon-1]==I[0]
        prob+= I[0]==0
        prob.solve(GUROBI())

    xlist=list()
    for v in prob.variables():
        if v.name[:1]=="x":
            y= v.name, "=", v.varValue
            xlist.append(v.varValue)
    ylist.append(xlist)

df= pd.DataFrame(ylist).transpose()
#writer = pd.ExcelWriter('X_Sol values for 9 clusters.xlsx', engine='xlsxwriter') #For 9
Clusters
writer = pd.ExcelWriter('X_Sol values for 962 clusters.xlsx', engine='xlsxwriter') #For 962
Clusters
df.to_excel(writer, sheet_name='Sheet1', startcol=0, startrow=0, header=False,
index=False)
writer.save()

print("--- %s seconds ---" % (time.time() - start_time))
```

MMR model:

```
import time
start_time = time.time()
from pulp import *
from openpyxl import load_workbook

#For 9 scenarios#
wb=load_workbook(filename= 'Compiled output.xlsx')
ws=wb['kmeans']

#For 962 scenarios#
#wb=load_workbook(filename= 'Price Curves 12-5-18.xlsx')
#ws=wb['Sheet2']

wb2=load_workbook(filename= 'X_Sol values for 9 clusters.xlsx') #For 9 Clusters
#wb2=load_workbook(filename= 'X_Sol values for 962 clusters.xlsx') #For 962 Clusters
ws2=wb2['Sheet1']

M=1000000
Cap=1000000
cin=305000
cout=-457500
d=1
Horizon=24
N=9

T=list(range(Horizon))
S=list(range(N))

p=[[0 for t in range(Horizon)] for s in range(N)]
x_sol=[[0 for t in range(Horizon)] for s in range(N)]

for t in range(Horizon):
    for s in range(N):
        p[s][t]=ws.cell(row=t+6, column=s+38).value   #For 9 scenarios
#         p[s][t]=ws.cell(row=t+2, column=s+2).value  #For 962 scenarios
for t in range(Horizon):
    for s in range(N):
        x_sol[s][t]=ws2.cell(row=t+1, column=s+1).value  #For 9 scenarios
#         x_sol[s][t]=ws2.cell(row=t+1, column=s+1).value   #For 962 scenarios
prob= LpProblem("EL-NR", LpMinimize)
x= LpVariable.matrix("x", (T), None, None, LpContinuous)
I= LpVariable.matrix("I", (T), 0, None, LpContinuous)
r= LpVariable('r',  0, None, LpContinuous)

prob+= r
```

```
for s in S:
    prob+=lpSum([-p[s][t]*x[t] for t in T]) + r >= lpSum([-p[s][t]*x_sol[s][t] for t in T])

for t in T:
    prob+=cout<=x[t]
    prob+=x[t]<=cin
    prob+=I[t]<=Cap
    if t>0:
        prob+=I[t]==I[t-1]+x[t-1]

    prob+= I[Horizon-1]+x[Horizon-1]==I[0]
    prob+= I[0]==0
    prob.solve()

print(prob.objective.value())

for v in prob.variables():
    if v.name[:1]=="x":
        print(v.name, "=", v.varValue)

print("--- %s seconds ---" % (time.time() - start_time))
```

7. DDP model:

```
import time
start_time = time.time()
from openpyxl import load_workbook
import pandas as pd
import numbers

wb=load_workbook(filename="C:\Fourth Term @ Dal\Project\Directive studies\DDP\DDP no
ratchet\data_noratchet.xlsx", data_only=True)
ws=wb['1000']

##for 1000 step size
x=806
s=1001
n=24

P=[0 for k in range(n)]
for k in range(n):
    P[k]=ws.cell(row=k+2, column=2).value

X=[0 for j in range(x)]
for j in range(x):
    X[j]=ws.cell(row=j+2, column=3).value
```

```
S=[0 for i in range(s)]
for i in range(s):
   S[i]=ws.cell(row=i+2, column=4).value

Sin=0
Sout=0

F=[[0 for j in range(x)] for i in range(s)]

#@@@@@@@@@@@@@@@@@@@@@ n=23
@@@@@@@@@@@@@@@@@@
class c:
   def abc1(self):
      self.df_output1 = pd.DataFrame()
      for count, k in enumerate(range(n)):
         for i in range(s):
            for j in range(x):
               if k==n-1:
                  if (S[i]+X[j])==Sin:
                     F[i][j]=-X[j]*P[k]
                  else:
                     F[i][j]="NA"
      self.Fbar=list()
      self.Xbar=list()
      for f in F:
         try:
            FFF=max([x for x in f if isinstance(x, numbers.Number)])
            XXX=X[f.index(max([x for x in f if isinstance(x, numbers.Number)]))]
            self.Fbar.append(FFF)
            self.Xbar.append(XXX)
         except ValueError:
            FFF="NA"
            self.Fbar.append(FFF)
            self.Xbar.append(FFF)
      self.df_output1["n="+str(k).format(k)] = self.Xbar
      print(dict(zip(S,self.Fbar)))

#@@@@@@@@@@@@@@@@@@@@@ 22>=n>=1
@@@@@@@@@@@@@@@@@@@@@
   def abc2(self):
      list2=(list(range(n))[::-1][1:n-1])
      self.df_output2 = pd.DataFrame()
      for count, k in enumerate(list2):
         for i in range(s):
            for j in range(x):
               try:
                  if max(S)>=(S[i]+X[j])>=min(S):
```

```
                    FFFFF=S[i]+X[j]
                    F[i][j]=-X[j]*P[k]+dict(zip(S,self.Fbar))[FFFFF]
                if max(S)<(S[i]+X[j])<min(S):
                    F[i][j]="NA"
            except TypeError:
                F[i][j]="NA"
        self.Fbar=list()
        self.Xbar=list()
        for f in F:
            try:
                FFF=max([x for x in f if isinstance(x, numbers.Number)])
                XXX=X[f.index(max([x for x in f if isinstance(x, numbers.Number)]))]
                self.Fbar.append(FFF)
                self.Xbar.append(XXX)
            except ValueError:
                FFF="NA"
                self.Fbar.append(FFF)
                self.Xbar.append(FFF)
          self.Fdict=dict(zip(S,self.Fbar))
          self.df_output2["n="+str(k).format(k)] = self.Xbar


#@@@@@@@@@@@@@@@@@@@@ n=0
@@@@@@@@@@@@@@@@@@@@@@@@@@

    def abc3(self):
        self.df_output3 = pd.DataFrame()
        for count, k in enumerate(range(n)):
            if k==0:
                for i in range(s):
                    for j in range(x):
                        if S[i]==Sin and max(S)>=(S[i]+X[j])>=min(S):
                            FFFFF=(S[i]+X[j])
                            F[i][j]=-X[j]*P[k]+dict(zip(S,self.Fbar))[FFFFF]
                        else:
                            F[i][j]="NA"
                self.Fbar=list()
                self.Xbar=list()
                for f in F:
                    try:
                        FFF=max([x for x in f if isinstance(x, numbers.Number)])
                        XXX=X[f.index(max([x for x in f if isinstance(x, numbers.Number)]))]
                        self.Fbar.append(FFF)
                        self.Xbar.append(XXX)
                    except ValueError:
                        FFF="NA"
                        self.Fbar.append(FFF)
                        self.Xbar.append(FFF)
```

```
            self.df_output3["n="+str(k).format(k)] = self.Xbar

#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@

    def abc4(self):
        writer = pd.ExcelWriter('output.xlsx', engine='xlsxwriter')
        self.df_output1.to_excel(writer, sheet_name='Sheet1', startcol=0, header=True,
index=False)
        self.df_output2.to_excel(writer, sheet_name='Sheet1', startcol=1, header=True,
index=False)
        self.df_output3.to_excel(writer, sheet_name='Sheet1', startcol=n-1, header=True,
index=False)
        writer.save()

#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@

    def abc5(self):
        wb=load_workbook(filename="output.xlsx", data_only=True)
        ws=wb['Sheet1']
        X=[[0 for i in range(s)] for k in range(n)]

        Xlist=list()
        Slist=list()
        Plist=list()
        for k in range(n):
            for i in range(s):
                X[k][i]=ws.cell(column=24-k, row=i+2).value

            if k==0:
                Xstar=max([x for x in X[k] if isinstance(x, numbers.Number)])
                Sstar=Sin+Xstar
                Gain=-Xstar*P[k]
                Xlist.append(Xstar)
                Slist.append(Sstar)
                Plist.append(Gain)
                print("X"+str(k)+":", Xstar)
                print("-----------------------")
            else:
                Xstar=X[k][S.index(Sstar)]
                Sstar=Sstar+Xstar
                Gain=-Xstar*P[k]
                Xlist.append(Xstar)
                Slist.append(Sstar)
                Plist.append(Gain)
                print("X"+str(k)+":", Xstar)
```

```
        print("-----------------------")
    print("Profit:",sum(Plist))


foo=c()
foo.abc1()
foo.abc2()
foo.abc3()
foo.abc4()
foo.abc5()
print("--- %s seconds ---" % (time.time() - start_time))

#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@
```