Agent driven Procurement platform using Reinforcement learning

Alekya Saladi
Artificial intelligence and machine learning
Great Lakes Institute of Management
Chennai, India
alekyasaladi04@gmail.com

Narmada Krishnasamy
Artificial intelligence and machine learning
Great Lakes Institute of Management
Chennai, India
narmada.krishnasamy@gmail.com

Satish Panchapakesan
Artificial intelligence and machine learning
Great Lakes Institute of Management
Chennai, India
satishpanchami@gmail.com

***Abstract*— In this paper we have implemented an agent driven optimization approach using reinforcement learning, in context to supply chain management. It aims on proving that the cost of procurement is optimized when building a model using reinforcement learning framework over traditional models.**

The Cost of Goods sold (COGS) in a typical process manufacturing setup is about 60% of the Net proceeds on Sale (NPS), so even a 1% improvement, for say, a billion-dollar enterprise would mean about 6 Mn savings in raw material cost through smarter sourcing decisions.

Keywords—supply-chain, cost optimization, reinforcement learning, procurement, machine learning

I. Introduction

The global sourcing for electrical and electronic component market is expanding at an exponential rate, both in breadth and width of OEM's owing to trends around protective tariff's, macro-economic shifts et al – all this is making it challenging for the Industry downstream who consume these OEM products to meet supply to demand, putting them into intense and competitive cost pressures. It has become imperative to have innovative sourcing strategies, which can be enable smarter decision making. Advanced algorithms can aid such decisions. This is foundational to an optimised supply-chain model that gets constantly re-evaluated by enterprises for both cost and sourcing resilience.

Critical challenges towards building a world class supply chain network is in implementing complex business goals, handling high level of uncertainty in supply-demand and also having to handle numerous decision-making variables and constraints. A well-designed supply-chain model should help the companies: to improve their margins, to aid them in making expansions into newer markets and to significantly decrease their production cost.

Supply-chain optimization is the application of processes and tools to ensure the optimal operation of a manufacturing and distribution supply chain. This includes the optimal placement of inventory within the supply chain, minimizing cost including manufacturing costs, transportation costs and distribution costs. The process often involves the application of mathematical modelling techniques using computer science.

This optimization approach we are building can have its application in all industries manufacturing and/or distributing goods, including retail, industrial products and consumer packed products. The focus of this paper is on optimizing the supply demand relationship between a production house and their supply sources available globally. .

.

II. Problem formulation

In today's competitive and dynamic market place, supply-chain decision makers in addition to their experience and intuition are forced to make use of all technological advancements to stay ahead in their decision-making process. On the advent of technologies like machine learning it has opened up various approaches and methods towards making successful sourcing decisions.

In order to build an artificial intelligence (AI) enabled supply chain product the initial analysis was to study a traditional product to identify any opportunity of improvement. Analyses were performed on a product sourcing platform called MeRLIN a product of Rheinburke. It is a direct sourcing strategic solution that offers seamlessly integrated sourcing process automation along with Supplier Relationship Management and planning functions. The products key role was in enhancing collaboration between buyers and suppliers through a value-driven sourcing strategy.

After analysis of the product, their traditional analytical tool was proposed to be replaced with an AI driven agent which will handle the procurement process with increased cost savings, expanded global sourcing, shorter turn around, risk mitigation and enhanced ROI. Since traditional methods perform exhaustive search of the input attributes and the approach of "optimization" is lost. Whereas ML models perform fast in production since they are pre-trained.

This paper establishes that agent driven optimization approach using more advanced Deep learning techniques is much more cost effective than other traditional optimization methodologies such as Linear Programming / Dynamic programming based tools.

III. Approaches and Application

In order to benchmark the solution outcome we originally modelled the problem in the lines of linear programming (simplex algorithm) with the objective function of minimizing the cost of procurement for a given demand quantity with a set of constraints. Subsequently we built a reinforcement learning model, training it to identify the right supply sources with the objective of minimizing the procurement cost.

A. *Approach 1 - Simplex Method*

1) *Theory and definitions*

Simplex algorithm is the classical method to solve the optimization problem of linear programming. Mathematically, linear programming optimizes (minimizes or maximizes) the linear objective of several variables subject to the given conditions/constraints that satisfies a set of linear inequalities. The first step towards formulating a linear function involves defining the following in the problem.

a) Decision variables are the quantities needed to solve the objective function. These variables are the decision makers of the objective function.

b) Objective function is profit function that maximizes or minimizes. It is the objective of making decisions.

c) Constraints are a set of restrictions or situational conditions to limit the value of the decision variables.

d) Optimal solution is one of the feasible function where the objective function is maximum or minimum.

e) Non-negativity restrictions the decision variables should always take non-negative value in any linear programming, meaning the decision variables are always greater than or equal to zero.

2) *Application*

To solve this approach we used PuLP a modelling framework for solving linear programming written in python. Modelling process for PuLP involves initializing the model, defining decision variables, defining the objective function, defining the constraints and solve model.

.

B. *Approach 2 – Reinforcement learning model*

1) *Theory and definitions*

Reinforcement learning (RL) framework is a type of dynamic programming that trains an agent with a set of rewards and punishment based on its interaction with its environment. RL is a powerful, self-adaptive optimization mechanism that fits well with the continuous changing environment. Formulating a business problem into Markov decision process (MDP) is the most challenging part of this model.

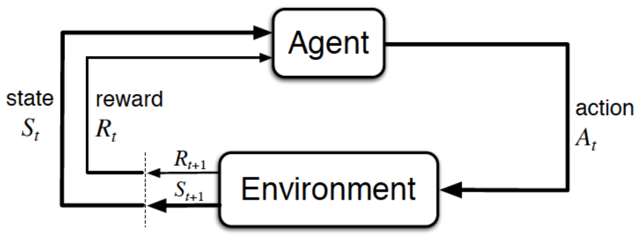

Fig. 1. Reinforcement learning model

At each time step, the agent receives the environment's state (the environment presents a situation to the agent), and the agent must choose an appropriate action in response. One time step later, the agent receives a reward (the environment indicates whether the agent has responded appropriately to the state) and a new state is presented to the agent**.** All agents have the goal to maximize expected cumulative reward, or the expected sum of rewards attained over all time steps. The important representations in the RL model for an agent are as follows:

a) State(S) represents the decision making factors which affects the reward, as observed by an agent in the operating environment.

*b) Action(*A*)* represents the agent's action which affects the states, so the agent learns to take the optimal action to maximize the reward.

c) Reward(R) represents the positive or the negative effects of an agent's action on its environment in the previous time instant.

d) Discount factor(γ) is used to balance immediate and future reward. For larger values of γ, the agent cares more about the distant future. Smaller values of γ result in more extreme discounting, where (in the most extreme case) agent only cares about the most immediate reward.

e) Learning Rate(α) determines to what extent newly acquired information overrides old information.

Q-learning is a popular technique in RL implemented with the Bellman equation Eq(1), the decision epoch is represented as $t \in T = \{1, 2, \ldots\}$, the knowledge obtained by an agent for a particular state-action pair at a time t is represented by a Q – function as follows:

$$Q_{t+1}(s_t, a_t) \leftarrow (1-\alpha)\, Q_t(s_t, a_t) + \alpha\, [r_{t+1}(s_{t+1}) + \gamma \max_{a \in A} Q_t(s_{t+1}, a)] \qquad (1)$$

where

a) $s \in S$ represents the state.

b) $a \in A$ represents the action.

c) $r_{t+1}(s_{t+1}) \in R$ represents the delayed rewards which is received at time t+1 for an action taken at time t.

d) $0 \leq \gamma \leq 1$ represents the discount factor. The higher the value of γ the greater the agent relies on the discounted future reward $\gamma \max_{a \in A} Q_t(s_{t+1}, a)$ compared to the delay reward.

e) $0 \leq \alpha \leq 1$ represents learning rate. The higher the value of α, the greater the agent relies on the delayed reward $r_{t+1}(s_{t+1})$ and the discounted future reward $\gamma \max_{a \in A} Q_t(s_{t+1}, a)$ compared to the Q- value $Q_t(s_t, a_t)$ at time t.

2) *Application*

Modelling an effective Rl solution includes:

a) Defining the state space .
b) Defining the initial state condition, reward / penality
c) Defining the Q-table (state – action)
d) Q-learning and Updating the table
e) Production and validation

IV. Data Structure

The dataset used in this paper includes the following attributes comprising of electrical, electronic and mechanical component parts sourced from global suppliers obtained from MeRLIN.

- Supplier ID
- Part Number (one to many Supplier-Part relationships)
- Supplier Quality rating by part
- Supplier sourcing tenure part
- Plant of Destination (one to many Supplier-Plant-Part relationships)
- Delivery Lead time by part
- Freight terms or Incoterms (set of 13 terms considered as Ordinal values)
- Minimum Order Quantity by Supplier ID
- Price per unit (normalized to a single currency)

For the purpose of modelling the supplier ID, Plant of destination and incoterms has been combined to create an attribute called Source ID. Since all the three attributes have many to many relationships the three attributes are combined together to create a unique identification named as Source ID
$a \in A$

Source ID = “Supplier ID” – “Plant of Destination” – “Incoterms”

For each Source ID an upper bound value is computed, which is the maximum quantity a Source ID can supply. Main purpose of computing this value is to mitigate risk involved in choosing a specific supplier. This maximum quantity is decided based on a set of constraints which are purposely applied in a hierarchical fashion as follows:

1) If the age of the supplier in the records is more than a specific threshold value (in this model its considered as 2 years) then the supplier is set to supply a maximum of 50 % of the demand otherwise the supplier can supply upto 30% .

2) Quality rating of the Supplier if its less than 3 then that Source ID is can supply upto 25 % of the demand.

3) Source ID whose delivery lead time is more than 3.7 weeks can supply only upto 10% of the supply value.

These Threshold values were decided based on inputs from the procurement decision makers of the traditonal model.

V. Data Modelling

A. Simplex algorithm

LP problem can be framed as maximization or a minimization problem in our case it is a cost minimization problem. The objective of the LP model is to minimize the cost of the part procured which is done by identifying the right supplier based on a set of limitations.

First step in solving the aforementioned problem is towards defining the problem, and then is defining problem variables. Supply quantity and price are the two variables used. Supply quantity is given a upper bound value, the maximum quantity that can be supplied by a source. Third step is to frame the objective function.

Objective function = (Supply quantity * Price of the product).

Deciding on the constraints is the fourth step. The only constrain is that supply quantity cannot exceed the demand quantity. Final step is executing solver function of the PuLP which will solve the minimization problem and will print the demand plan for the procurement of one part number at a specific price..

This demand plan will give us the details about purchasing what quantity from which supplier will minimize the purchase cost at a minimal risk.

Now that we have the price at which a particular part can be purchased across the various supplier options at a minimal risk, our next step is to build a RL model and prove that the agent can optimize this procurement process and purchase can be done at a much lower cost.

B. Reinforcement learning model

1) *State table*

The state is defined as a three dimensional vector with each element represented as follows

a) Source Id

b) Remaining demand quantity – is the remaining demand to be supplied at any point.

c) Remaining source quantity – is the remaining quantity a source can still supply at any point.

To generate the state table for each part following steps are followed.

a) Iterate over each Source Id initializing the first value to the maximum demand quantity of the part and maximum quantity a source can supply. Each state is identified with the combined values of source id, remaining quantity and source quantity.

Initial State = 'Source id' – 'Demand quantity' – 'Source quantity'

b) Next state is calculated by decrementing the demand quantity and source quantity by the minimum order quantity. Each iteration will give a new state with the source id, remaining demand quantity and source quantity.

Initial State = 'Source id' – 'Remaining demand quantity' – 'Remaining source quantity'

c) Iteration terminates when the remaining demand quantity or the remaining source reaches to a value equal to or less than the minimum order quantity. This state is called the "Absorbing state". Keep track of all the absorbing states, since the q-table updation starts from the very same states.

d) These above steps are repeated for each source Id of a particular part. The same steps are followed for creating the state table for all the parts.

2) *Q – Table*

When Q-learning is performed we create what's called a q-table or matrix that follows the shape of [state, action]. With respect to this problem the actions taken by an agent is to choose between supply and no supply.

a) First step is to get the absorbing states of all the source id in order. Iteration starts from the absorbing states for a particular source Id.

b) Initialize the q-table values to zero.

c) On each iteration (episode) the q-table is updated using the Bellsman equation. Eq (1). In order for the update to happen the following values are computed.

- Q-value of that particular state is initially 0.
- α (alpha) which is the learning rate is given a constant value of 0.01
- r_{t+1} is the reward or the penalty that is for taking a particular action. This calculation is explained in detail under the sub heading Rewards.
- The q-value gets updated as the states move from one to another, the max q value of the next state is added to the reward of the current state q value.
- Incrementing the demand quantity with minimum order quantity gives a value which is the reference to identifying the next state. From the current state we move to the next state which refers this value in the state name.

d) The above calculation is done and the q-table is updated with the q values for both actions of supply and no supply for every state starting from the first absorbing state.

e) From the current state we move to the next state by identifying the state name with the incremented value of the demand quantity and the remaining source quantity with the minimum order value.

f) The loop for the absorbing state goes on till the remaining demand quantity equals to the demand and the remaining source quantity equals to the maximum supply quantity of that source.

g) The same iteration happens for all the absorbing state of the source id and the whole process repeats itself for all the source id's of a particular part.

The table (TABLE-I) is the state action table or the Q-table which has the two action Supply and No supply mapped to the state name. Each state is constructed as mentioned above and their q-value for that particular action is calculated with the Bellsman's equation explained in detail in the next section.

TABLE I. Q-TABLE SAMPLE

State	Supply	No supply
M0006412-6861010-4-299558-149779	88.935273	37.929309
M0006412-6861010-4-269602-119823	84.287354	33.281391
M0006412-6861010-4-239646-89867	73.958647	22.952683
M0006412-6861010-4-209690-59911	51.005963	0.000000
M0006412-6861010-4-269602-149779	88.935273	37.929309
M0006412-6861010-4-299558-119823	84.287354	33.281391
M0006412-6861010-4-299558-89867	73.958647	22.952683

This q-table becomes a reference table for our agent to select the best action based on the q-value..

3) *Rewards*

A reward is received after completing certain action at a given state. This value is computed based on whether the particular part can be supplied by the source id or not for the two actions of supply and no supply as follows

a) Part supplied by the source id

If that source id supplies the part then we construct a 3 –bit binary string as a weightage to the reward function. It is constructed as follows:

Binary weight =

[rating_indicator, delivery_indicator, age_indicator]

- *Rating indicator* - This bit turns on when the quality rating of the supplier > 3
- *Delivery indicator* - This bit gets turned on when the delivery lead time is supplier < 3.7 weeks
- *Age indicator* - This bit gets turned on when tenure of the supplier > 2 years

The priority of the bit decides the weightage to be added to the reward or penalty which depends on the action taken.

Supply _weight = Integer value of the binary weight

The reason for computing the reward and penalty with the supply weight is to give addition weightage to supplier based on their indicators.

For example: if the quality and the delivery time of a supplier is good, even if he is a new supplier (supplier with less than 2 years' experience in the market) can be considered with a higher weightage than a supplier with lesser turn-around time and more experience.

```
if price < base price:
    if action= supply :
        reward = supply_weight*abs(((base_price- price)/
        base_price)*100)
    if action = nosupply:
        penality = -compliment(supply_weight)
else :
    if action= supply :
        penality = -compliment(supply_weight)
    if action = nosupply:
        reward = compliment(supply_weight)*abs((
        (base_price - price)/ base_price)*100)
```

b) Part not supplied by the source id

```
if action = supply :
    penality = -0.0001
if action = nosupply :
    reward= 10
```

4) *Production*

a) Drop the records wherever the no supply q-value is greater than the supply q- value. Remaining records are sorted w.r.t to the supply q- values in descending order .

b) For a particular part the price at which the source id supplies the part is included as a column in the table w.r.t the source id to calculate the cost. The table is sort w.r.t to the price column in ascending order. When there are two source id with the same q-value then the source with the lowest price is chosen.

c) By keeping check on the demand quantity of a part, first the supplier with the maximum supply q- value is chosen and the corresponding supply quantity is considered. This value is subtracted from the demand quantity of the part.

d) The above step is repeated with the next highest supply q-value until the demand quantity becomes zero. When the loop is repeating another condition is checked so that the supplier quantity does not exceed the maximum supply quantity of that supplier.

e) Next step is to calculate the supply cost for each source id with the total supply quantity.

Cost per supplier = Supplier selling price * Quantity from the supplier as suggested by the algorithm.

f) Sum all the cost per supplier values to get the final optimized cost at which the particular part can be procured.

VI. Conclusion and Future Work

Reinforcement learning and linear programming model has been applied to optimize the procurement price of a part. Linear programming has proved to have limited capabilities when compared to a RL model which can handle risk mitigation by choosing the right suppliers very well.

TABLE II. Result Table - Output Comparison

Part Number	**Price**	
	LP	*RL*
P521002BFALX00	95.26	85.0
P147045BFDMCAD	137.80	137.80
P610007BFAAX00	1689.75	1596.32

Price of a part obtained from both the models

The above table shows the illustrative results from both linear programming (LP) model and reinforcement (RL) model for a sample set of parts. Results for samples show improved results in the RL approach.

The price optimisation was better to the extent of 8% in the best case while it showed a marked improvement or neutral for a few parts.

In summary, RL optimization techniques have showed a great potential to solve supply chain management problems. Certainly there is great deal of improvement in this RL model which can still decrease the procurement cost.

Acknowledgment

The authors are grateful for the extensive support provided by Dr. D. Narayana mentor at Great Lakes Institute of Management Chennai.

.

References

[1] R. Sutton and A. Barto, Reinforcement Learning , MIT Press., Cambridge, MA., 1998

[2] Business Statistics – A first course by PK Viswanathan

[3] Pierpaolo Pontrandolfo, Abhijit Gosavi, O Geoffrey Okogbaa, and Tapas K Das. Globalsupply chain management: a reinforcement learning approach.

[4] Fisher M.L. (1997) 'What is the right supply chain for your product', Harvard Business Review

[5] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, , , , , , and , Reinforcement Learning in Different Phases of Quantum Control, arXiv:1705.00565.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors, , , , , and , Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors, arXiv:1207.0580

[7] R. Al-Rfou et al., Theano: A Python Framework for Fast Computation of Mathematical Expressions, , theano: A python Framework for Fast Computation of Mathematical Expressions, arXiv:1605.02688.

[8] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Optimal Control of Coupled Spin Dynamics: Design of NMR Pulse Sequences by Gradient Ascent Algorithms, J. Magn. Reson. 172, 296 (2005).

[9] AlphaGo Zero Is Not A Sign of Imminent Human-Level AI' Skynet Today.