Agrilogistics - A Genetic Programming Based Approach

Divya D. Kulkarni(✉) and Shivashankar B. Nair

Indian Institute of Technology Guwahati, Guwahati, India
{divyadk,sbnair}@iitg.ac.in

Abstract. The advent of technology in the agriculture sector, such as precision agriculture, the Internet of Things (IoT) and machine learning has dramatically improved the experience of farming scenario. Apart from improving the farming conditions, there is a need for focused effort to achieve a balanced ecosystem in the supply chain of agrilogistics. Inefficient price signals conveyed to the farmer, erratic price fluctuations and inflation of the agri-produce coupled with the presence of several intermediaries, tend to imbalance the system. In this work, we propose an IoT based agrilogistic system coupled with a genetic programming algorithm to ensure fair prices across all the participants within. The system evolves and generates a set of programs that, in turn, generates the selling rate for every participant in the supply chain in a manner that confers fairness.

Keywords: Internet of Things (IoT) · Genetic Programming · Agrilogistics · Supply chain

1 Introduction

The agriculture sector has seen enormous changes lately due to the advent of technology. An IoT plays a significant role in traditional precision agriculture, wherein the data is collected from a variety of connected devices and sensors, setup in the fields, thus improving both yield and productivity. IoT in agriculture mainly comprises setting up sensors across large tracts of fields, surveying and collating data mostly using either sensor networks or drones. Sensors used are varied in nature and include those that can sense temperature, humidity and soil-moisture [4]. Cameras have also been used as sensors to judge the quality of the plants [1]. Various solutions have been proposed to augment IoT with precision agriculture [3,9,11,14], but only a few have been actually implemented effectively [6,18]. Many have proposed the use of machine learning and data mining techniques for improving yield, controlling pests, managing soil, etc. [15,16] but actual implementations of only a few have shown decent results [5,7].

There are several other factors that need to be addressed in the making of a fair agriculture based ecosystem. For instance inefficient price signals, the

P. Pereira et al. (Eds.): SC4Life 2019, LNICST 318, pp. 83–96, 2020.
https://doi.org/10.1007/978-3-030-45293-3_7

presence of too many intermediaries and information asymmetries [2,12] can lead to a severe losses to a farmer.

In order to protect the producer from erratic price fluctuations and to handle the problem of inefficient price signals in agrarian countries like India, the government generally sets up a Minimum Support Price (MSP).

Though this may protect the farmers from incurring severe losses, it does not guarantee the best price for their produce mostly because they are not aware of the overall dynamics of the market. Presence of too many intermediaries such as middle-men, agents, and brokers between the point of production (farm) to the point of vending and consumption also causes erratic price fluctuations. The problem at times can cascade resulting in a gross wastage of the product due to high production and low demand. Governments also ensure to stock warehouses managed by them with the produce to control market trends. Such warehouses buy the produce from farmers at the MSP and endeavour to always maintain a buffer stock of the produce. It may happen that the quantum of produce stored exceeds the buffer stock. Under such conditions, the same is released to the market at a fair price. Such an increase or decrease in stock in the warehouse coupled with dynamically varying production and demands can create a large turbulence in profits made by the supply chain. Mau et al. [10] points out that the agriculture sector lacks efficient consumer response, which could guide the producer in the production of the goods based on the demands of the customer. It is reported in [17] that profits are mostly skewed away from the producers due to the presence of various intermediaries. In an agrilogistic system, ensuring a mechanism that profits all the participants is an uphill task. To handle this supply chain in agrilogistics (logistics system of the agriculture sector), this mechanism would need to be aware of all the connected dynamics of the market so as to advise each of the participants of the action to be taken by them.

One of the methods to solve the supply chain agrilogistics is to initially collect a huge amount data which naturally can be generated only over a large period of time, and then analyze it using various machine learning techniques including deep neural networks. This could mean huge losses (during the data collection interval) with hardly anything gained or learned. A mechanism wherein learning commences during the data collection phase itself would greatly ameliorate the problem. One of the ways this could be realized is by making programs evolve using Genetic Programming (GP) [8] based techniques during the course of data collection. Execution of such programs will help steer the system being controlled to a better level. In GP, the quality of a program is measured by a fitness score, which in an agrilogistic scenario could be the selling rate of the product at every stage. In addition, an IoT based agrilogistic supply chain system, data received from heterogeneous sensors could guide the evolution of such programs.

In this work, we propose a Genetic Programming (GP) [8] based strategy to learn to handle the dynamics of the supply chain and accordingly advise the participants on whom to sell their stock in a network of producers (farmers), warehouses and vendors so as to get the best price. The strategy takes into consideration the dynamically changing demand, the quantum of stock,

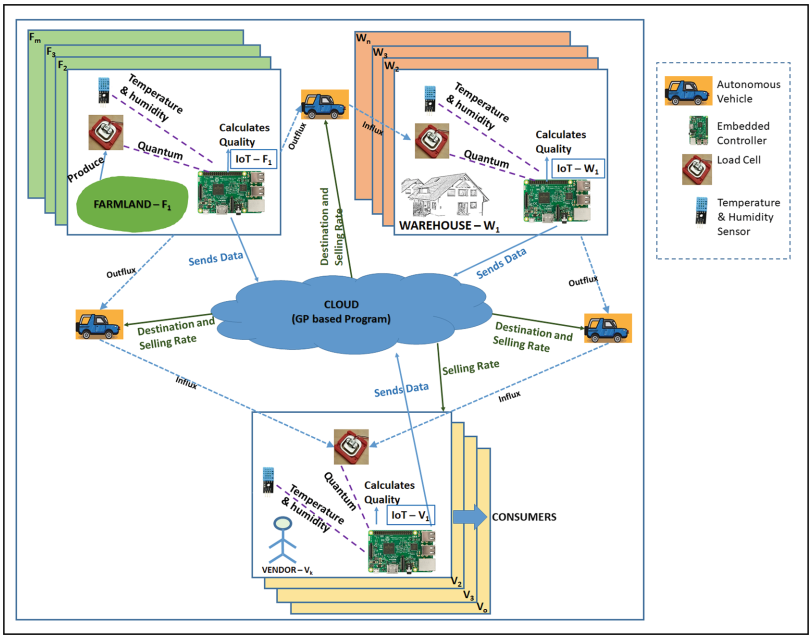

Fig. 1. The proposed model of supply chain management in agrilogistics

temperature, humidity and time elapsed after harvest and evolves programs over time which when run maximizes the selling rates across all the participants.

2 Methodology

The work herein describes a method to generate and evolve efficient programs for an IoT based supply chain of agrilogistics shown in Fig. 1. As illustrated the participants include sets of farmers, F $= \{F_1, F_2, F_3..., F_n\}$, warehouses, W $= \{W_1, W_2, W_3, ..., W_m\}$ and vendors, V $= \{V_1, V_2, V_3, ..., V_p\}$. A farmer F_i who generates the produce can either send them, part or whole, over to a warehouse W_j or a vendor V_k. Likewise W_j, which maintains a buffer stock of the produce, could channel excess amounts to any of the vendors based on demand. Eventually, the vendors sell the produce to the consumers. In addition, there are q number of number of autonomous vehicles (G) at a site x (where $x \in \{\{F_i\}, \{W_j\}, \{V_k\}\}$), G_{x_q}, at each of the locations or sites of the participants. A vehicle G is responsible for transporting a certain quantity of produce from one site to the other.

The IoT infrastructure is used to sense and record parameters, such as ambient temperature and humidity, age of the produce, its quantum and quality, at every site. The recorded data is then sent to the cloud which runs the proposed GP based algorithm to evolve programs, with the primary objective being to maximize the selling rates, while at the same time maintaining their consistency at the respective sites. The evolved program provides the destination where the produce has to be transported to and the selling rate at the destination.

Important Parameters. The main parameters and terms used in the system are explained below.

1. *Temperature (T) and Humidity (H):* Both temperature and humidity affect the quality of the produce. The measurements are done at the sites of every F_i, W_j and V_k using the temperature and humidity sensors at respective IoTs installed therein. Based on fixed predetermined ranges of values, temperature and humidity, are recorded as High (hi), Moderate (mod) and Low (lo).
2. *Quantum of Produce (Q):* It is the net quantity of a specific produce available at an instant of time at the site of a participant. Q also assumes values denoted as hi, mod and lo. It is measured using a load cell connected to the IoT at the respective sites.
3. *Age (τ):* The *age* of any produce is defined as the time elapsed from the time of harvest to the current time. The age of the produce increases irrespective of its location and is independent of all other parameters.
4. *Quality (ϕ):* The *quality* of the product affects its overall selling rate. Higher the *quality*, higher would be its selling rate. The quality of produce is assumed to degrade with increase in either T or H or τ. Just as temperature and humidity, ϕ of a produce takes values hi, mod and lo.
5. *Demand (δ):* It is one of the most significant terms and affects the dynamics of the system. δ increases when Q at the vendors is lo and decreases otherwise. Demand influences the production and outflux rates of the produce from a site, thereby affecting their selling rates at different sites that constitute the system. Demand also is expressed as hi, mod and lo.
6. *Distance (D) to the Destination (ψ):* It is the distance between the site from where the produce is to be shipped and the site where it is to be delivered (Destination site ψ).
7. *Outflux (O) of Produce:* Outflux is amount of produce a deliverer ships to the destination (ψ). It is calculated locally at the respective IoT site. In our work we assume the produce P to be *onions*,*tomatoes* and *cotton*.
   Outflux of the produce P from the site x to the site ψ is given by the formula below:
   $$O_P^{x \rightarrow \psi} = \alpha * \delta_P * Q_P^x \tag{1}$$
   where α is a constant and δ_P^x and Q_P^x are the demand and the quantum of the produce P at the site x, respectively and x and $\psi \in \{\{F\}, \{W\}, \{V\}\}$.
8. *Selling Rate (ρ):* The selling rate per unit of a produce, ρ, is determined at every site x for every produce P in the supply chain. The selling rate is directly proportional to the quality, ϕ, of the produce, demand from the customer, δ, and the distance, D, to the destination where it needs to be transported. It is inversely proportional to the quantum, Q, and age, τ, of the produce.

Selling rate, ρ, is given by the Eq. (2):

$$\rho_P^x = \beta * \frac{\delta_P * \phi_P^x * D_P^x}{Q_P^x * \tau_P^x} \tag{2}$$

where β is a constant, P is the produce whose ρ is being calculated at site x respectively, $x \in \{\{F\}, \{W\}, \{V\}\}$.

The parameters T, H and Q are sensed using the associated sensors while those of τ, ϕ, and O are computed by the respective IoTs, locally. The rest wiz. δ and ρ are computed in the cloud as shown in the Fig. 1.

Evolution of Programs. The generic version of a program in a population of programs, used in this work, has three tuples corresponding to a *farmer*, *warehouse* and *vendor* respectively, as shown below.

$$S \equiv ((F^{(T,H,P,\delta_P,\phi_P,Q_P,\psi,\rho_P)^i}), (W^{(T,H,P,\delta_P,\phi_P,Q_P,\psi,\rho_P)^j}), (V^{(T,H,P,\delta_P,\phi_P,Q_P,\rho_P)^k}))^l \tag{3}$$

where T, H, δ, Q, η can take either of the values *hi*,*mod* or *lo* and S is set of all such programs evolved. Each tuple of the program refers to all the *farmers*, all the *warehouses* and all the *vendors* at that instant of time. This tuple denotes the rule that infers the selling rate and the destination to which the produce needs to be shipped from the corresponding site i.e. F^i, W^j or V^k based on the values of the parameters. At the site, V^k, the produce is sold to the customers. For instance, the portion of the first tuple ($i = 1$) related to the first farmer's site of the program is illustrated below.

$$((F^{(lo,lo,onions,hi,hi,mod,W_2,45)})^1), (F^{(hi,lo,onions,hi,mod,mod,W_1,30)})^1)$$

The above portion of the program that can be expressed as a rule below -

FOR{*the site of the farmer* F_1} **IF** {*the temperature and humidity are low, the produce is onions with high demand having high quality and moderate quantum*}, **THEN** {*the selling rate is 45 units and the destination to which it needs to be shipped is* W_2} **ELSE IF** {*the temperature is high and humidity is low, the produce is onions with high demand having moderate quality and moderate quantum*} **THEN** {*the selling rate is 30 units and the destination to which it needs to be shipped is* W_1}.

Programs such as these are stored in a repository in a cloud. In the current scenario, the selling rate, ρ, of a produce is considered as the fitness score of the associated program. The goal is to maximize ρ so that the concerned seller get the maximum profit. It may be noted that the selling rate is contained by the demand of the consumers (as seen in Eq. 2) and hence programs evolve in a manner that is congenial to all participants, including the consumers.

The Algorithm

In the GP approach, the initial programs are generated randomly. In the work reported herein, it is proposed that the sensed parameters *T, H, Q and* ϕ are obtained and calculated using the IoT infrastructure at each of the sites participating in the agrilogistics scenario, thereby greatly curtailing the otherwise random search space. The GP program is assumed to be hosted in a cloud.

When all these values from all the sites are initially sent to the cloud, the demand of each type of produce is computed. Based on the demand, the outflux (O) of the produce is calculated. Using a randomly selected destination (ψ), the selling rate (ρ) of the produce (P) is then computed. These values viz. P, O, ρ and ψ are sent to the concerned site. These computations are done for every P at every site at the cloud. The respective IoT at the sites, communicates with an autonomous vehicle or a transporter within its site, which in turn delivers the O units of the produce P to the concerned destination, ψ. A program, as discussed earlier, which includes all the sites is thus generated and stored in a repository within the cloud. The cloud always receives the data from all the IoTs and structures the incoming data, I in the form shown below as a three tuple:

$$I \equiv (F^{\{T,H,P,\phi_P,Q_P\}^i}, F^{\{T,H,P,\phi_P,Q_P\}^j}, V^{\{T,H,P,\phi_P,Q_P\}^k})^l \tag{4}$$

The next time, the cloud receives and structures an I, the GP program first checks whether the first tuple for the first farmer, F_I^1, sent by the site matches with any one of the former portion of the first tuple F_S^1 in the set of stored programs S. If so, ψ_S^j (which could be either W_S^j or V_S^j) from the latter portion of F_S^1 is retrieved. Using this information, the GP program retrieves the tuple for the same in the second or third tuples, as the case maybe, in S. This retrieved tuple is compared with the corresponding tuple W_I^1 or V_I^1 (as the case maybe). The corresponding ψ_S^k (which could be either W_S^k or V_S^k) is retrieved. Using this information, the GP program retrieves the tuple for the same in the third tuples in S. This retrieved tuple is compared with the corresponding tuple W_I^1 or V_I^1 (as the case maybe). If all comparisons are true, then the three tuples from S namely F_S^1, W_S^j and V_S^k sent back to the IoT site of F_1. In case, no match is found, the GP evolves a new program using crossover and mutation as explained later.

The crossover operation is applied when the currently reported parameters match with at least one tuple in S stored in the repository. The operation of crossover involves two *parent programs* in S, the first *parent* is one whose former portion of the first tuple in S matches with the first tuple of I. The other parent in S could be one whose former portion of any of its two latter tuples matches with either of the second or third tuple of I. The crossover operation is illustrated in the Figs. 3, 4 and 5. If there are multiple candidates for the *parent programs*, then those *parents* are selected which infer the highest selling rates. The crossover operation results in two new programs are stored in the repository and retrieved as and when required in future. In this condition too, the three tuples from S namely F_S^1, W_S^j and V_S^k sent back to the IoT site of F_1.

If a few parameter values in F_I^1 do not match with the former portion of F_S^1, then those in F_I^1 are copied on to F_S^1 to achieve the mutated solution. Then, the corresponding selling rate is calculated and the solution is sent back to the IoT site of F_1. The mutation operation is illustrated in the Figs. 6 and 7.

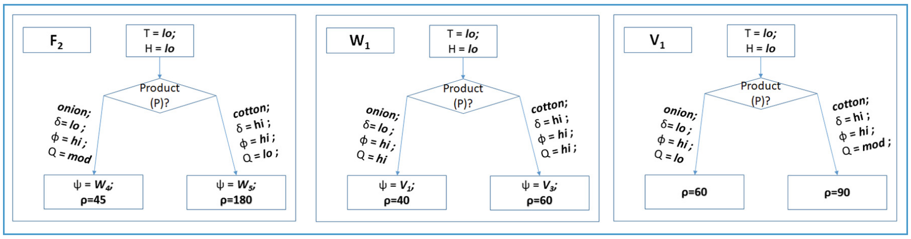
F₂
T = lo; H = lo
Product (P)?
onion; δ= lo; ϕ = hi; Q = mod
cotton; δ = hi; ϕ = hi; Q = lo;
ψ = W₄; ρ=45
ψ = W₅; ρ=180
W₁
T = lo; H = lo
Product (P)?
onion; δ= lo; ϕ = hi; Q = hi
cotton; δ = hi; ϕ = hi; Q = hi;
ψ = V₁; ρ=40
ψ = V₃; ρ=60
V₁
T = lo; H = lo
Product (P)?
onion; δ= lo; ϕ = hi; Q = lo
cotton; δ = hi; ϕ = hi; Q = mod;
ρ=60
ρ=90

Fig. 2. Two of the programs generated at the initial generations corresponding to the items - **onion** and **cotton**

The process of matching, crossover and mutation is done across all tuples in I so as to complete one generation of evolution.

If there are no matching programs to perform either of the crossover or mutation operations, then a new program is generated and routed to a randomly selected destination with the selling rate determined as per the Eq. 2.

3 Results

Generation of Initial Population of Programs. We simulated the GP cum IoT side algorithms while generating the appropriate values of the pertinent sensors. First, we generated the initial population of programs as described in the previous section, and stored them in the repository within the cloud. Figure 2 shows the illustration of two of the initial the programs generated as per Eq. 3 for the products (*onion*) and (*cotton*). There are three blocks in the program viz. the farmer, the warehouse, and the vendor blocks representing the three tuples in Eq. 3, each corresponding to the conditions observed at the respective sites. In Fig. 2, the ρ corresponds to the fitness of that particular program at that site for the associated produce. The selling rate is calculated as per the Eq. 2. We generated 50 such programs as the initial population and stored them in the repository in the cloud.

Evolving New Programs. After the initial programs were generated over time, when the sites report the next set of observed conditions to the cloud, the GP program searches the repository for the matching programs as explained in the previous section. If the matching program is found, the cloud reports the ϕ where the O_P^x has to be routed, and ρ_P^x, at which the produce has to be sold to the ϕ. If a match is not found, then either the operation of crossover or mutation is applied.

The Figs. 3, 4 and 5 depict the scenario, where crossover operation is applied. Parent 1 program in Fig. 3 represents *Parent* 1, where the tuple corresponding to the site of the farmer F_2 match with the currently reported parameters of F_2, whereas the remaining tuples corresponding to W_1 and V_2, do not match with the currently reported parameter values. Similarly, *Parent* 2 program in

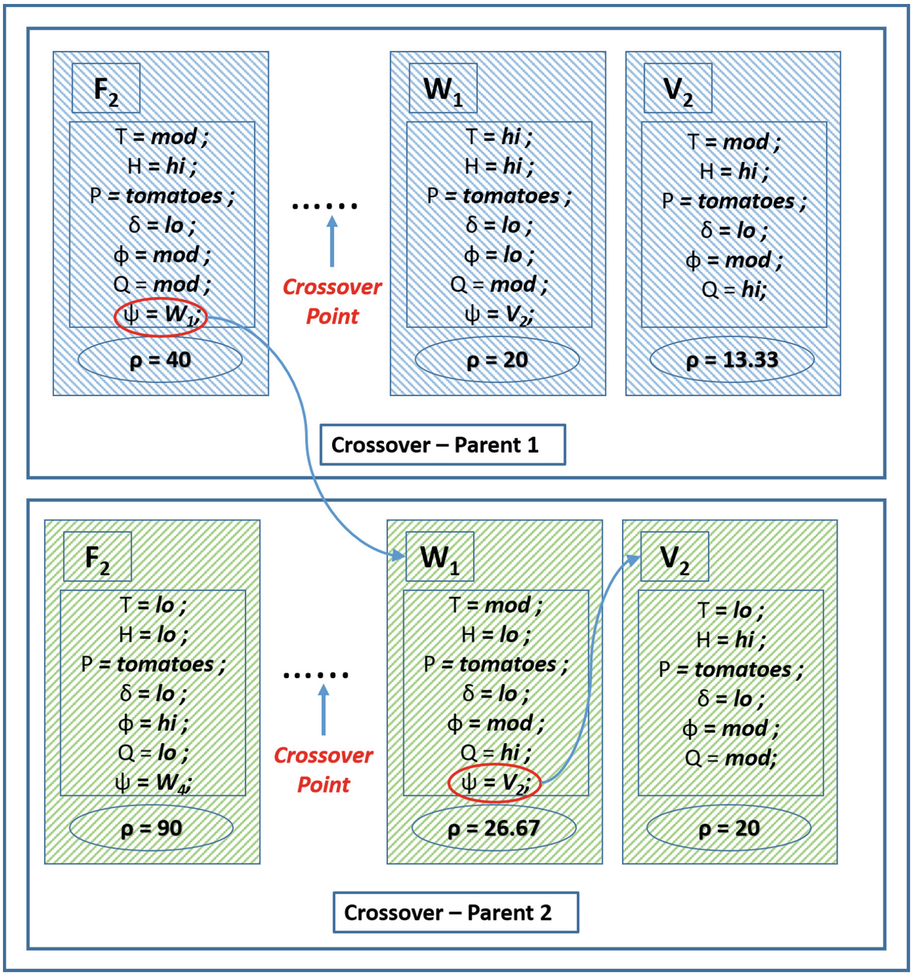

Fig. 3. Parent programs - crossover operation

the same Fig. 3 depicts *Parent 2* where the tuples corresponding to W_1 and V_2 match with the respective currently reported parameter values. The crossover mechanism is applied on the matched programs to obtain two new programs, one of them which can cater to the current parameters as shown in the Fig. 4 and the other one shown in Fig. 5 is saved in the repository for future use.

Mutation operation is depicted in Figs. 6 and 7. The *parent* program in Fig. 6, has the value of the parameter Q as *lo* for the site F_1, whereas the currently reported value of Q at F_1 is *mod*. This parameter is mutated as shown in the Fig. 7 and a new program is obtained. The selling rate (or the fitness) of this newly generated program is calculated and updated in the repository.

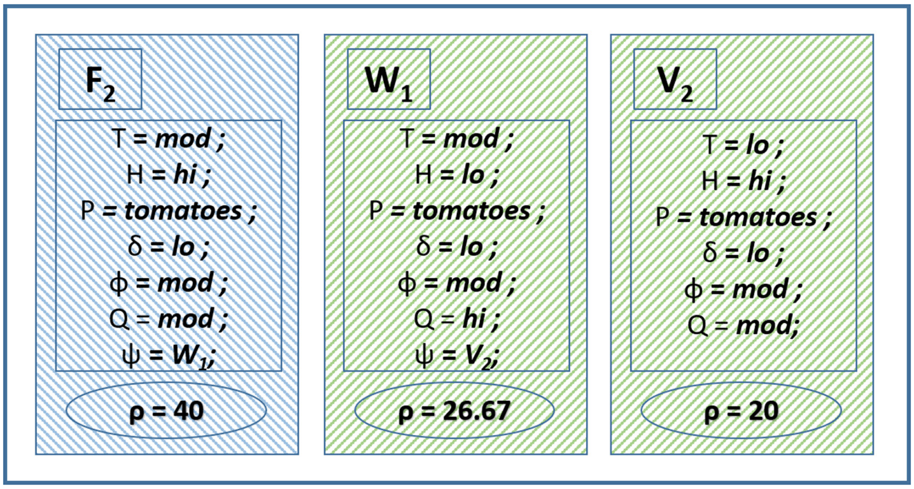

Fig. 4. Resultant program 1 - post crossover operation

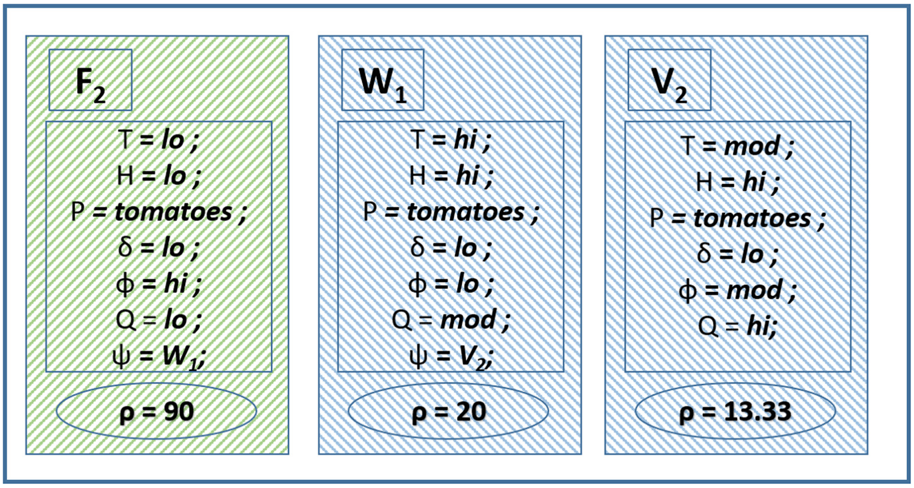

Fig. 5. Resultant program 2 - post crossover operation

As we generate and evolve the programs as per Eq. 3, the programs grow in dimension. The pseudocode in Pseudocode 1 and the Fig. 8, depicts the programs assembled for the site F_1 at an instant of time. As more and more programs are evolved, the conditions in the programs also increase, thereby resulting in a complex program. When we observe Figs. 2 and 8, we can observe the increasing conditions as more programs evolve.

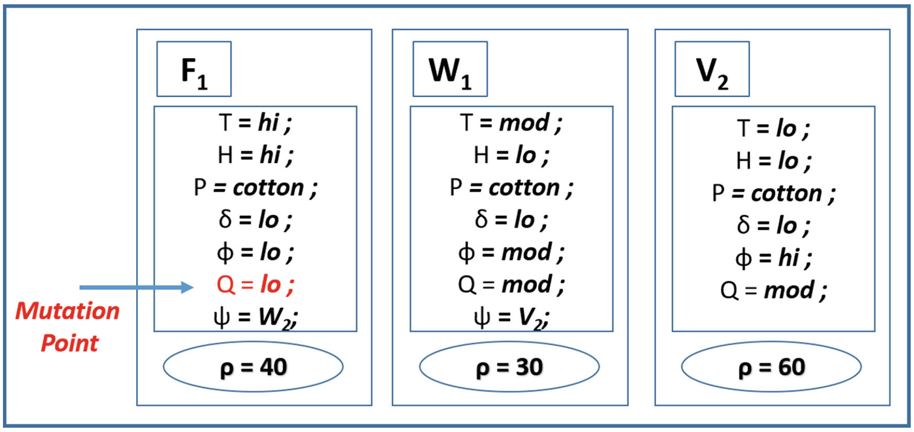

Fig. 6. Parent program - mutation operation

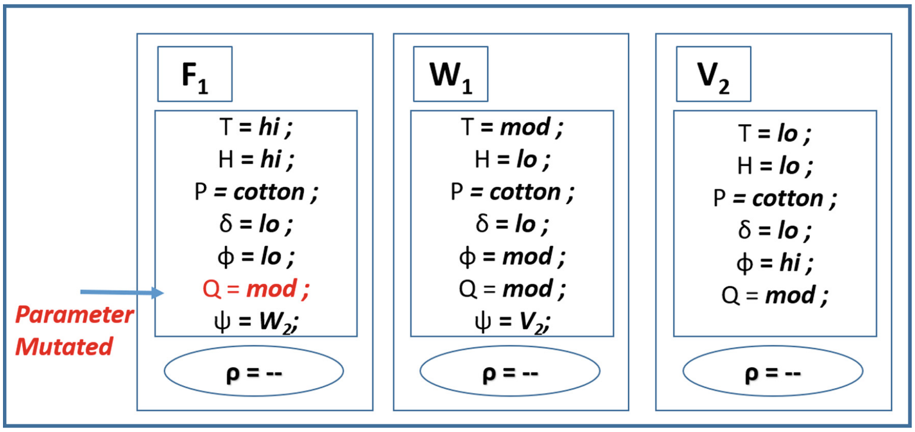

Fig. 7. Resultant program - post mutation operation

The graph of the selling rate of all the participants F, W and V plotted against the generations as the programs evolve is shown in the Fig. 9. The selling rates of all F, all W and all V is given by σ_F, σ_W and σ_V, respectively, as below:

$$\sigma_F = \sum_{i=1}^{m} \sum_{q=1}^{P} \rho_q^{F_i}$$
$$\sigma_W = \sum_{j=1}^{n} \sum_{q=1}^{P} \rho_q^{W_j} \tag{5}$$
$$\sigma_V = \sum_{k=1}^{o} \sum_{q=1}^{P} \rho_q^{V_k}$$

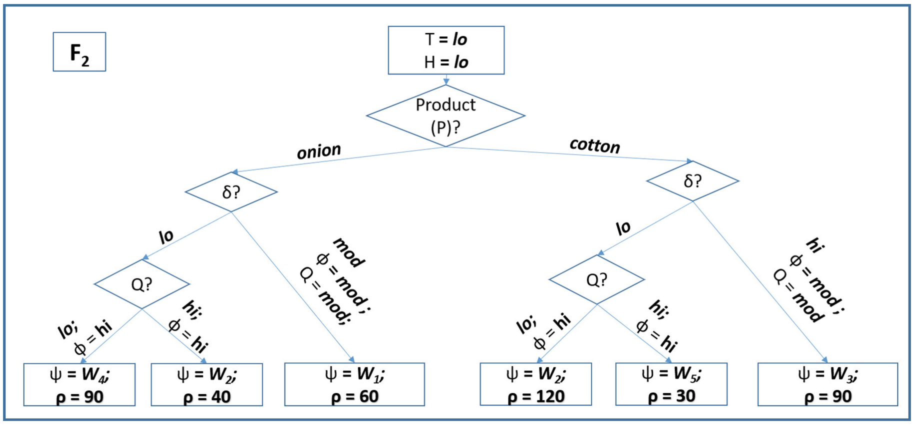

Fig. 8. A Snapshot of assembled programs

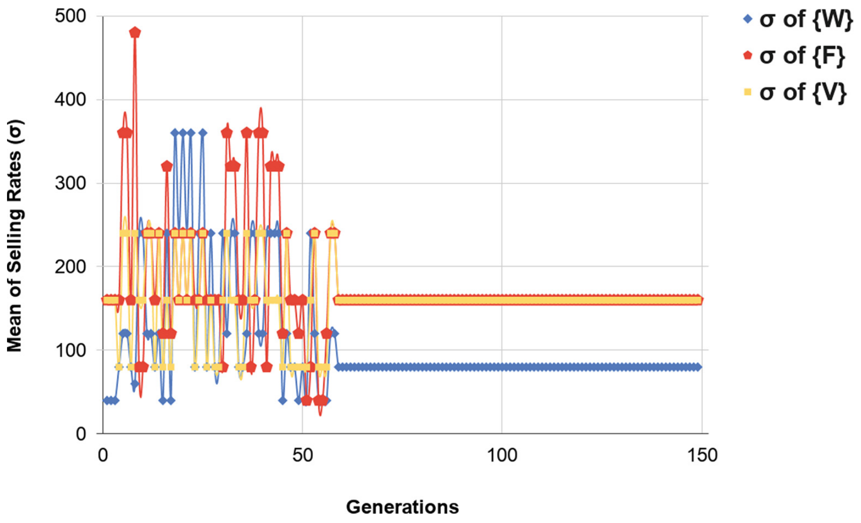

Fig. 9. Graph of the selling rates across generations

In the graph, shown in Fig. 9, the selling rates till 50^{th} generation represent the initial population of programs generated. After which, the programs evolve as per the GP based algorithm. In the initial generations, we can observe fluctuations in the σ_F, σ_W and σ_V, wherein only one of the three sets of participants (F, W and V) seem to get high selling rates whereas others are getting lower values. However, as the programs evolve, all the σ converges to narrow bands. As can be seen, σ_F and σ_V converge to almost the same narrow band, while σ_W converges to a lower value. This is because, at each W_i there is certain amount of buffer stocked, causing an increase in the respective Q, consequently decreasing the ρ as per the Eq. 2. This lower value of σ_W conforms to the standard

Pseudocode 1 : Part of the pseudocode of one of the programs assembled

```
switch $X$ do
    case $X \leftarrow F_2$ do
        if $T == lo$ && $H == lo$ then
            if $P == onion$ then
                if $\delta == lo$ then
                    if $Q == lo$ && $\phi == hi$ then
                        $\psi \leftarrow W_4$
                        $\rho \leftarrow 90$
                    else if $Q == hi$ && $\phi == hi$ then
                        $\psi \leftarrow W_2$
                        $\rho \leftarrow 40$
                else if $\delta == mod$ then
                    if $Q == mod$ && $\phi == mod$ then
                        $\psi \leftarrow W_1$
                        $\rho \leftarrow 60$
            else if $z == cotton$ then
                if $\delta == lo$ then
                    if $Q == lo$ && $\phi \leftarrow hi$ then
                        $\psi \leftarrow W_2$
                        $\rho \leftarrow 120$
                    else if $Q == hi$ && $\phi == hi$ then
                        $\psi \leftarrow W_5$
                        $\rho \leftarrow 30$
                else if $\delta == hi$ then
                    if $Q == mod$ && $\phi == mod$
                     then
                        $\psi \leftarrow W_3$
                        $\rho \leftarrow 90$
```

warehousing practices [13]. It can thus be seen that each participant in the supply chain gets a fair rate for the produce.

4 Discussions and Conclusions

We have proposed a novel way of looking at the Supply Chain in the agriculture sector, to setup fair prices of commodities by taking into consideration the factors which affect the selling rate. This will eliminate the influence of middlemen on the prices, and possibly bring down erratic price fluctuations. The proposed work combines an IoT with supply chain management in agrilogistics and gives an indication to the farmer as to what products are in demand enabling him/her to concentrate on producing them. This insulates the farmer from producing excessively large quantities of a low demand produce or vice versa which may otherwise result in wastage and financial losses. Such an end to end system, provides for a win-win situation for all the participants involved in the supply chain. This can be clearly seen from the resulting graph wherein the selling rate for all participants seem to converge to a narrow band. While the IoT provides a clear and exact picture of the ground truth of the produce at various sites, the GP learns from what it has made the system do in the past using the programs in its repository. In some sense it learns from scratch and does not really require for one to wait for the collection of large amounts of data before arriving at a solution. Most of the current AI methods bank on the initial availability of a huge amount of relevant data which is then churned to produce some meaningful results using computationally heavy algorithms such as deep learning. Learning from what has been done so far during the data collection phase is grossly missing. The paradigm discussed herein could thus be well suited for learning during a data collection phase. It may even allow for better and more pertinent data to be generated thereby eventually increasing the efficacy of the traditionally used large-data driven algorithms. There are yet certain other aspects which need to be addressed to optimize the transport of goods from one site to another as also put the IoT in the real-world. Our future work will thus include the realization of an actual agent based IoT and cloud to comprehend the challenges of the paradigm when implemented in the real-world.

References

1. Bock, C.H., Nutter Jr., F.W.: Detection and measurement of plant disease symptoms using visible-wavelength photography and image analysis. Plant Sci. Rev. **6**, 73 (2012)
2. Chand, R.: Development policies and agricultural markets. Econ. Polit. Wkly. **47**(52), 53–63 (2012). http://www.jstor.org/stable/41720551
3. Gebbers, R., Adamchuk, V.I.: Precision agriculture and food security. Science **327**(5967), 828–831 (2010). https://doi.org/10.1126/science.1183899. https://science.sciencemag.org/content/327/5967/828
4. Gondchawar, N., Kawitkar, R.: IoT based smart agriculture. Int. J. Adv. Res. Comput. Commun. Eng. **5**(6), 838–842 (2016)
5. He, Y., Zeng, H., Fan, Y., Ji, S., Wu, J.: Application of deep learning in integrated pest management: a real-time system for detection and diagnosis of oilseed rape pests. Mob. Inf. Syst. (2019). https://doi.org/10.1155/2019/4570808
6. Kapetanovic, Z., Vasisht, D., Won, J., Chandra, R., Kimball, M.: Experiences deploying an always-on farm network. GetMobile Mob. Comp. Commun. **21**(2), 16–21 (2017). https://doi.org/10.1145/3131214.3131220
7. King, A.: Technology: the future of agriculture. Nature **544** (2017). https://doi.org/10.1038/544S21a
8. Koza, J.R., Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection, vol. 1. MIT Press, Cambridge (1992)
9. Mat, I., Kassim, M.R.M., Harun, A.N., Yusoff, I.M.: IoT in precision agriculture applications using wireless moisture sensor network. In: 2016 IEEE Conference on Open Systems (ICOS), pp. 24–29. IEEE (2016)

10. Mau, M.: Supply chain management in agriculture - including economics aspects like responsibility and transparency. In: European Association of Agricultural Economists, 2002 International Congress, 28–31 August 2002, Zaragoza, Spain (2002)
11. Mekala, M.S., Viswanathan, P.: A survey: smart agriculture IoT with cloud computing. In: 2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS), pp. 1–7, August 2017. https://doi.org/10.1109/ICMDCS.2017.8211551
12. Negi, S., Anand, N.: Supply chain of fruits & vegetables' agribusiness in uttarakhand (India): major issues and challenges. J. Supply Chain Manag. Syst. **4**(1), 43–57 (2015)
13. Ozsen, L., Coullard, C.R., Daskin, M.S.: Capacitated warehouse location model with risk pooling. Naval Res. Logistics (NRL) **55**(4), 295–312 (2008)
14. Patil, K.A., Kale, N.R.: A model for smart agriculture using IoT. In: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), pp. 543–545, December 2016. https://doi.org/10.1109/ICGTSPICC.2016.7955360
15. Patil, S.S., Thorat, S.A.: Early detection of grapes diseases using machine learning and IoT. In: 2016 Second International Conference on Cognitive Computing and Information Processing (CCIP), pp. 1–5. IEEE (2016)
16. Shekhar, Y., Dagur, E., Mishra, S., Sankaranarayanan, S.: Intelligent iot based automated irrigation system. Int. J. Appl. Eng. Res. **12**(18), 7306–7320 (2017)
17. TeamYS: Agri-logistics in India: challenges and emerging solutions (2013). https://yourstory.com/2013/05/agri-logistics-in-india-challenges-and-emerging-solutions/
18. Vasisht, D., et al.: FarmBeats: an IoT platform for data-driven agriculture. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pp. 515–529 (2017)