# Precision Farming

Mechatronic Systems Engineering

Prof. Dr. Stefan Henkler


Abhinandan Dinakar
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
Email:
abhinandan.dinakar001@stud.fh-dortmund.de

Nikhil Ganapathy Manjapura
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
Email:
nikhil.ganapathymanjapura001@stud.fh-dortmund.de

Rushab Jahagirdar
Embedded Systems Engineering
Fachhochschule Dortmund
Dortmund, Germany
Email: rushab.jahagirdar001@stud.fh-dortmund.de



***Abstract*— Determining the cost-benefit of precision agriculture management has always proven to be difficult. Many of the technologies in use today are still in their infancy, making it difficult to pinpoint prices for equipment and services. This allows one to explore better techniques and equipment for the advancement of agriculture management. Precision farming is an attempt to approach real time management, focused on observation, measurement, and response to variability in crops, fields, and surroundings. It helps improve crop yields, reduce costs, including labor costs, and optimize process inputs. All of this helps improve profitability. Parallelly, precision agriculture increases worker safety, reduces the environmental impact of farming and farming practices, and contributes to the sustainability of agricultural production.**



***Keywords*— *Precision architecture, Path finding approach, Smart farming, Decision tree, Modelling techniques.***


## I. Introduction

This paper summarizes the design of bale collector swarming in precision farming. To increase the agricultural yield and to ease the process, rising technology of robotic swarming plays very important role. Swarm of robots can be used in various agriculture applications, from planting the seeds, spraying pesticides, monitoring crop health to harvesting crops.

With the less human interaction, large area cultivation is possible and consumes less time than manual work would take. All the bale collectors in the swarm are efficient when they cooperate in the group. Parallelism in swarming- in which the task is divided into sub tasks and can be allocated to different robots in the swarm. Since this technology is autonomous, robots can cope with any environmental changes.

*Figure 1. Swarming Example*

There are some disadvantages of this swarm technologies. Since it is completely autonomous, there can be some fatal accidents due to system shutdown, or algorithm failure. Also, battery optimization is also one of the key features that need to be considered in the design. Initial investment for this technology will be very high.

In this paper, basic design of the bale collector, and the algorithm for picking and dropping of the bale with appropriate UPPAL state diagrams are discussed. This paper also summarizes how role of the robot in the swarm is decided using decision tree algorithm.

## II. Vehicle Design

The below Fig. 2, shows the design of the bale collector. This design varies with size for different types of bales. This vehicle is designed to be more energy efficient and has very few active components. In this design there is one motor (1) to drive a conveyor to pick up the bale. One motor (2) to drive the rear wheel and front wheels are belt driven. One hydraulic lifter (3) lifts the roller conveyor platform (4) for unloading.

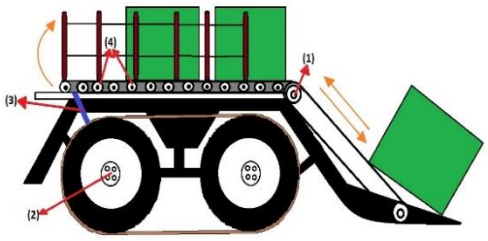


*Figure 2. Vehicle Design*

When loading, vehicle's inclined front part picks the bale, and the motorized conveyor moves the bale to the non-motorized roller conveyor platform. When picking up the next bale, the first bale in the front on the platform is pushed back by the next bale.

When unloading, hydraulic lifter lifts the roller conveyor platform, and the motored conveyor rotates in clockwise. All the bales slide down to the ground from the vehicle. Since rear wheels are belt driven, it makes the vehicle feasible to all terrain.

Ultrasonic sensors are used for obstacle detection and camera is used for bale detection. Vehicle to vehicle communication is established using local TCP network.

## III. CONSENS MODEL

### A. *Requirements Diagram*

The requirements are derived with respect to CONSENS model. Fig. 3 shows the Electrical Requirements for the system. The primary requirement should be that the vehicle is battery operated. The other sub-requirements are like Charging, Back Up and Power supply.

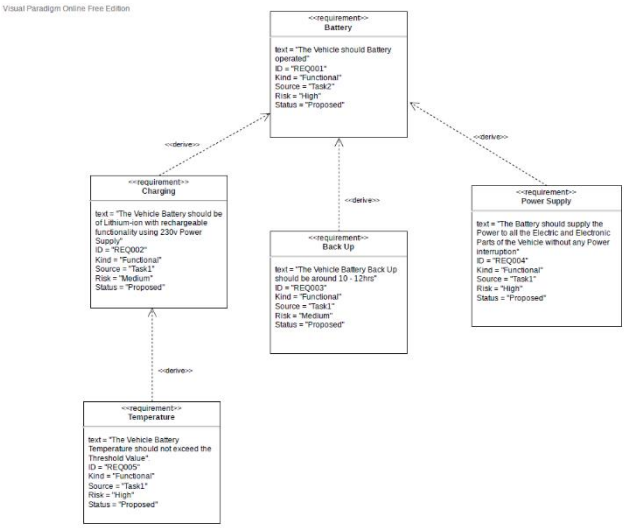


*Figure 3. Electrical Requirements of the System*

The vehicle should use Lithium-ion battery and it should be rechargeable is the Charging requirement. While charging the battery, its temperature of the battery should not exceed threshold value is other derived requirement for Charging.

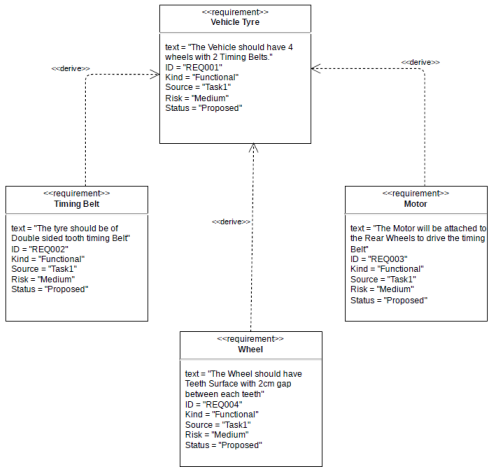


*Figure 4. Mechanical Requirements of the System*

Fig. 4 shows the Mechanical Requirements of the system. The main requirement is "The Vehicle should have 4 wheels with 2 timing belts". The sub-requirements derived are Timing Belt, Motor and Wheel related.

### B. *Software Hierarchy*

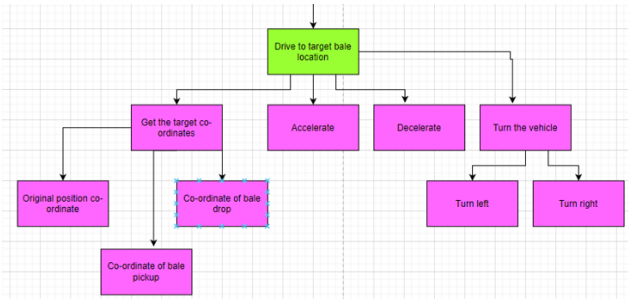


*Figure 5. Drive to Target location*

The complete software hierarchy for the Bale Collection Vehicle is divided in to four parts namely Drive to target bale location, identifying the bale, pick the bale and self-diagnostics.

The above fig. 5 shows the software hierarchy for driving to the target location. It contains the sub hierarchies like acceleration, deceleration, turning the vehicle and getting the coordinates of the bale.

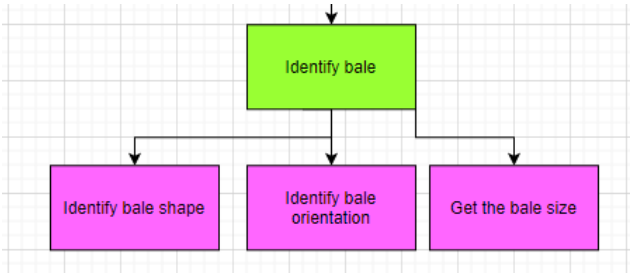


*Figure 6. Bale Identification*

Fig. 6 shows the hierarchy for identifying the bale. It focuses on identifying the bale shape, identifying bale orientation and getting the bale size.

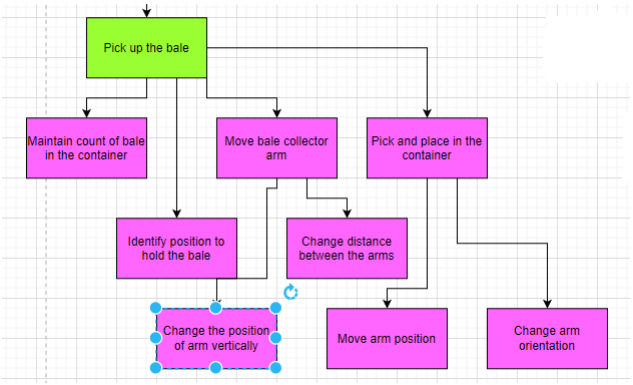


*Figure 7. Picking the Bale from location*

Picking the bale from its location will be done in four different steps. It will maintain the count of bales in the container, move bale collector arm and pick and place in the

container and identifying position to hold the bale. Fig. 7 shows the hierarchy for the further steps.

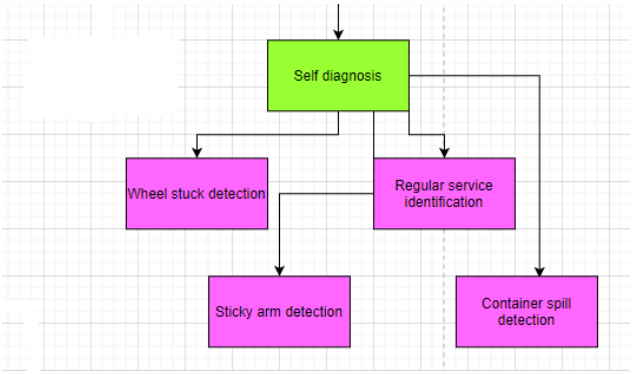


*Figure 8. Self-Diagnosis*

Fig. 8 shows the Self-Diagnosis of the vehicle system. Self-diagnosis will be done for wheel stuck detection, sticky arm detection, container spill detection and regular service identification.

### C. Component Diagram

This below fig. 9 represents the component diagram which shows the dependencies of other module on the system. In this system, bale coordinates are acquired by a field camera.

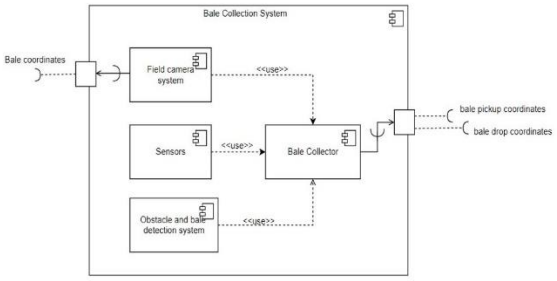


*Figure 9. Component Diagram of the system*

Modules like field camera system, sensor system, obstacle and bale detection system are the dependencies of bale collector system.

## IV. CONTROL REQUIREMENTS

- The co-ordinate of the bale is received by the system which alerts the system to move towards those co-ordinates and collect the bale.
- When the system reaches the destination i.e., bale. The system stops receiving the co-ordinates, this signal acts as feedback to the system indicating that it is the right time to pick up the bale.
- A loading arm is fixed in front of the bale truck which picks up the bale and places it in the loader.
- When the camera no longer detects any bale, it acts as feedback to the system indicating it should move back to the initial point.
- After placing the bales in the loader, the bale truck moves back to the initial state.
- The bale is then emptied with the help of a conveyor belt.
- An ultrasonic sensor is mounted on the truck. This output indicates if there are any intrusion. This acts as feedback to the system.
- Positive feedback – If there is no intrusion. Indicating the system to keep moving in the same direction
- Negative feedback – if there is intrusion. Indicating the system to change its direction.

Control strategy - A monitoring station can continuously monitor the system. Adversely if the system fails to perform as desired. The system can be turned off from the monitoring station.

## V. REAL TIME STATE CHARTS (RTSC)

Fig. 10 shows the real time state chart for Camera module. Once the camera gets connected then it will start with the configuration.

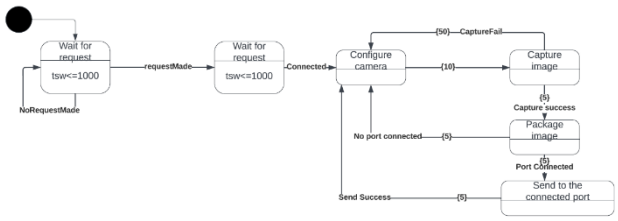


*Figure 10. RTSC for Camera*

The camera will capture image, image packaging, and send to the connected port. If the system fails, then the feedback will be given to configure camera.

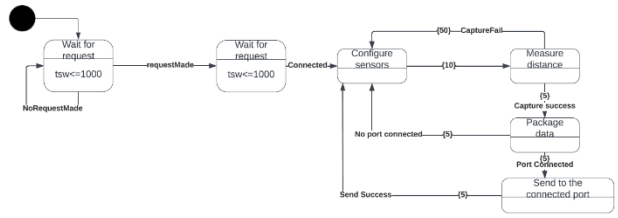


*Figure 11. RTSC for Sensor*

The above fig. 11 shows the real time state chart for sensor. The measurement of distance will be in a packed data and it will be sent to a connected port. If any failure in the system will give feed back to the configure system block.

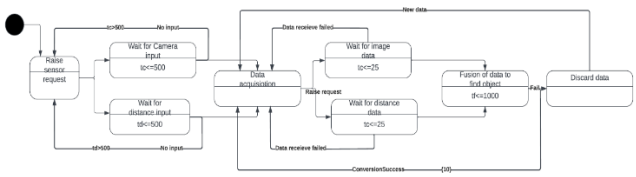


*Figure 12. RTSC for Sensor and Camera data fusion*

Fig. 12 shows the fusion of both camera and sensor data. The data acquisition is the process of converting real world physical conditions to digital numeric.

## VI. UPPAAL MODELS

Uppaal is a modeling tool environment for validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.) [4].

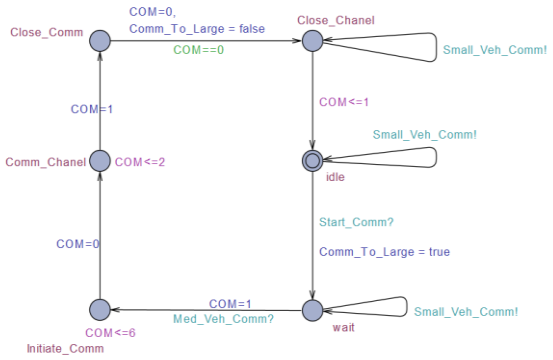


*Figure 13. Communication between Vehicles*

Fig. 13 explains about Communication between the Vehicles. The Large Vehicle will be the Manager Vehicle. There are 6 states in this system namely idle, wait, initiate_Comm, Comm_Chanel, Close_Comm and Close_Chanel.

The Medium and Small Vehicle will communicate with Large Vehicle on a single communication channel. Whenever the Medium Vehicle is communicating with Large Vehicle then at that time Small Vehicle cannot communicate with Large Vehicle.

The Small Vehicle will wait for the Medium Vehicle to finish the communication and close the communication channel, then Small Vehicle gets the chance to communicate.

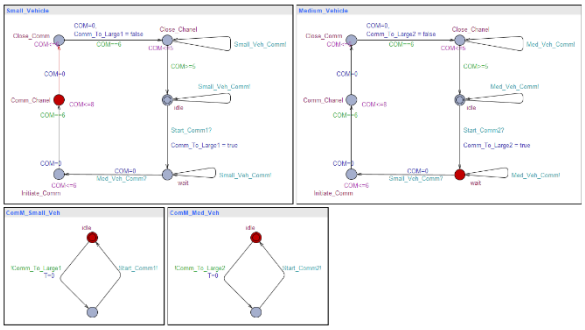

*Figure 14. Simulation for Communication*

Fig. 14 shows the Simulation for Communication between the Vehicles. Any one of the vehicles can start the communication. It will start from the idle state and move to wait, if any other vehicle is using the communication channel. If the channel is free then it will initiate the communication.

Once the data is successfully transferred then it will close the communication and its channel. When the Communication and channel closing takes place, then at that time other vehicle can start the communication.

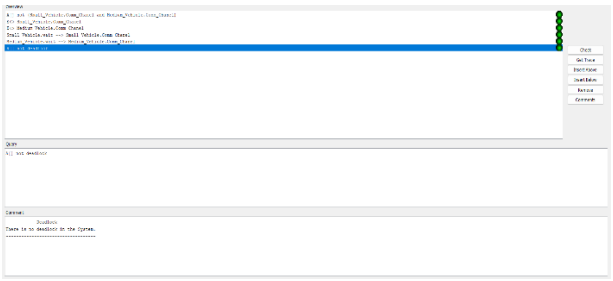

*Figure 15. Verification for Communication*

Different type of verifications confirm that the system is stable. Fig. 15 shows different verifications for the communication behaviour of the system.

The verification part confirms that there is no deadlock in the system, both small and medium size vehicle cannot access the communication channel at the same time and so on. Here all the properties are satisfied.

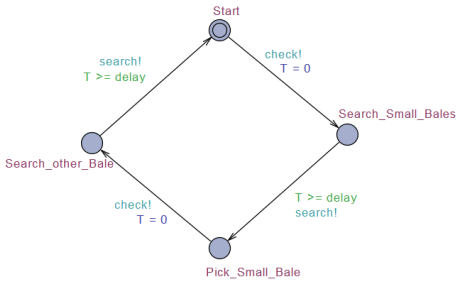


*Figure 16. Small Bale picking by Medium and Large Vehicle*

In above Fig. 16, we can see the behavior of picking a small bale by a large and medium size vehicle. When the Small Vehicle has no storage space to pick small bale then the medium and large vehicles pick the small bales accordingly.

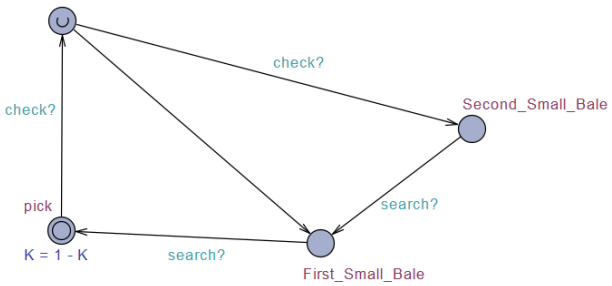


*Figure 17. Small Bale pickup assignment*

Whichever vehicle (Large or Medium) gets assigned for picking up the small bale will start moving towards that small bale and picks it up. In fig. 17, it will check whether both the small bales are picked or not then according to that the Large and Medium Vehicles move to pick those bales.

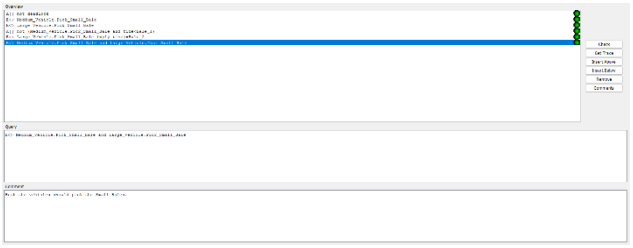

*Figure 18. Verification for Small Bale pickup system*

Fig. 18 shows the different type of verifications performed for Small Bale collection system to check the system behavior. The different type of verifications are that there should be no deadlock, Medium vehicle and Large vehicle can pick the small bales and both vehicle has access to small bale picking.

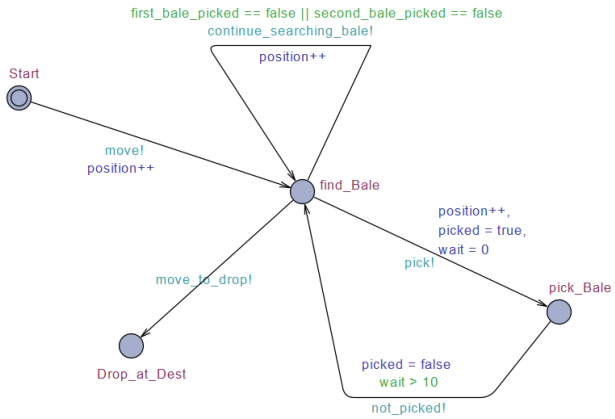


*Figure 19. Two Medium Bale collection by medium vehicle*

In the above system, the behavior of medium vehicle collecting two medium size bales can be seen. The vehicle starts searching medium bales and it will pick the first assigned bale and followed by the second assigned bale. Once both the bales are collected then the vehicle moves to destination point.

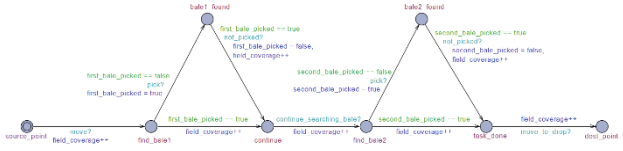

*Figure 20. Route map for Medium Bale Collection*

In fig. 20, the Medium Vehicle will start from source point and move to find the first bale. If it finds the first bale then it will move to pick the first medium bale else it will move to find the second bale. Once all the bales are picked then it will move to destination point to drop the bales.

## VII. ALGORITHM FOR PATH DETECTION

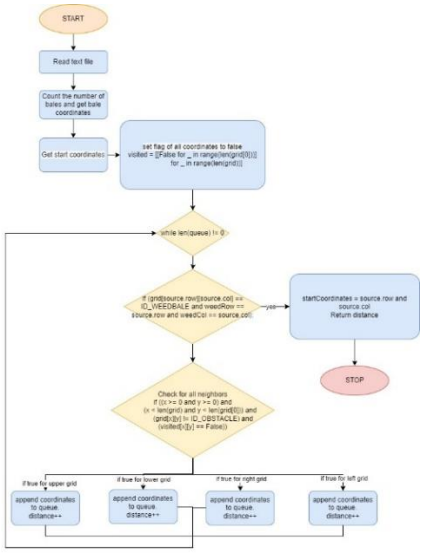


*Figure 21. Path detection algorithm*

The flow chart shows the algorithm used to determine the path to weed bales. Same algorithm is used to determine the path of other types of bales. Initially, this algorithm reads the text file from the directory and counts the number of different types of bales. Coordinates of different types of bales are also stored in an array.

In the beginning of the algorithm, start position coordinates are set. From start grid, its neighbors are checked, if there is no obstacle, if that grid is within the boundary conditions and if that grid is not visited before.

If it satisfies all the conditions, the coordinates of that grid are appended in the queue with the step count. And now all the neighbors of that grid are checked for all conditions and the valid grid coordinates are stored in the queue. This process repeats until the grid coordinates is equal to bale coordinates which means bale is detected.

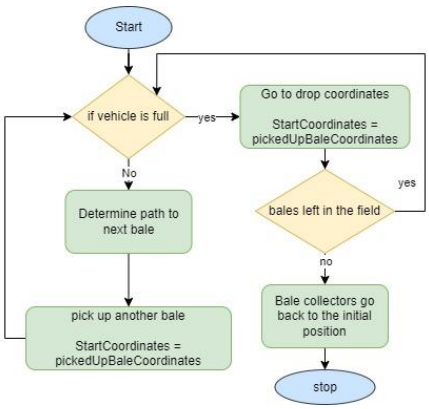


*Figure 22. Pick and drop algorithm*

Once the bale is detected, the bale is picked up and the coordinates where the bale was picked up becomes the start coordinates and goes to pick up the next bale until the vehicle is full.

Once the vehicle is full, vehicle goes to last bottom right grid and drops all the bales as shown in the fig.22. If there are any

more bales left in the field, drop point is updated as start point, and vehicle determines the path to the leftover bale and pick up and drops it at the drop destination.

Once all the bales in the field are picked up and dropped at the drop point, vehicles go back to initial position and goes into idle state.

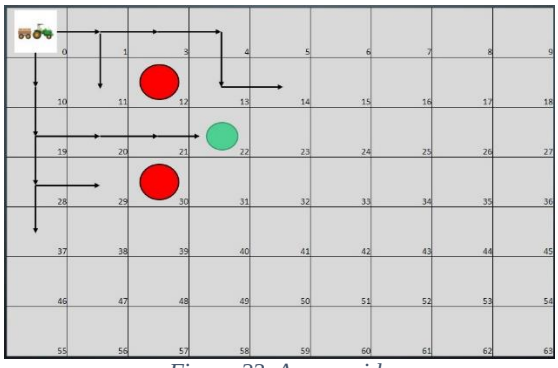

*Figure 23. Arena grids*

| r | c | d |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 2 | 0 | 2 |
| 1 | 1 | 2 |
| 0 | 2 | 2 |
| 3 | 0 | 3 |
| 2 | 1 | 3 |
| 0 | 3 | 3 |
| 4 | 0 | 4 |
| 3 | 1 | 4 |
| 2 | 2 | 4 |
| 1 | 3 | 4 |
| 0 | 4 | 4 |
| 5 | 0 | 5 |
| 4 | 1 | 5 |
| 2 | 3 | 5 |
| 1 | 4 | 5 |
| 0 | 5 | 5 |
| 5 | 1 | 6 |
| 4 | 2 | 6 |

*Figure 24. Queue to store valid grid coordinates and step count*

Above grids shows the arena where grid-0 is the start point and grid-22 is where the bale is. Grid 12 and grid-30 are the obstacles indicated in red circles. From grid-0, neighbors are checked for obstacles, boundary conditions and previously visited. Valid grid's coordinates are stored in the queue as shown in the fig. 24.

Queue will have all the valid coordinates (r, c) and the path must be determined using back path tracing algorithm. Queue is read from the back. From destination for each step count (d) valid neighbor with the step count (d-1) is determined and it is stored in another final path queue. From grid with step count (d-1) next valid neighbors is found with step count (d-2) and stored in the final path queue.

This step repeats until d=1 and source coordinates are reached. Now by reversing the final path queue, we can determine the path from one point to another. Which vehicle must pick up which bale? – this vehicle role will be decided using decision tree algorithm which will be explained in the next part of the paper.

## VIII. OUTPUT

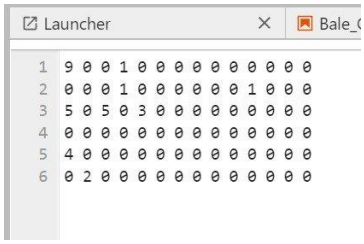


*Figure 25. Input arena*

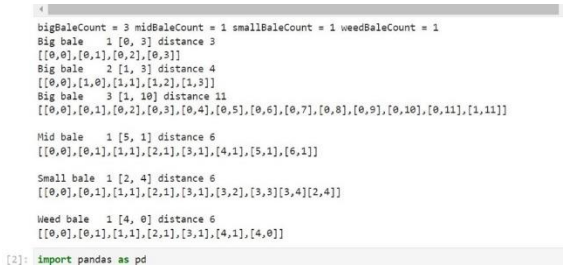


*Figure 26. Path detection algorithm output*

In this program, arena text file shown in the figure 25 is imported. We have made the following assumptions-

| Sl. No. | Parameters | Code |
|---|---|---|
| 1 | Starting Point | 9 |
| 2 | Big bale | 1 |
| 3 | Medium size bale | 2 |
| 4 | Small size bale | 3 |
| 5 | Weed bale | 4 |
| 6 | Obstacle | 5 |
| 7 | Valid path | 0 |

When the algorithm is executed, it prints number of different types of bales and coordinates of the path to that bale from source.

## IX. DECISION TREE ALGORITHM

### A. *Introduction*

The success of machine learning techniques applied to financial and medical problems can be hampered by inherent noise in the data. If noise is not properly accounted for, the data can be overfitted, producing unnecessarily complex models and leading to misinterpretations. Therefore, many efforts have been made to improve the interpretability of models in machine learning applications.

Decision Trees (DT) are popular machine learning models applied to both classification and regression tasks using well-known training algorithms such as CART, C4.5, and Boosted Trees. DT is considered an explainable model because it has fewer nodes than other node-based models. Additionally, tree structures can return output with significantly less computation than other more complex models. However, as explained, greedily constructed trees are unstable. To improve stability, new algorithms use tree ensembles such as bagging trees, random forests (RF), and XGBoost (XG). However, increasing the number of trees also increases the number of nodes, thus complicating the model.

A decision tree is a classification and prediction tool with a tree-like structure where each inner node represents a

test for an attribute, each branch represents the result of the test, and each leaf node (terminal node) contains a class label. There is a small decision tree above. A key advantage of decision trees is their high interpretability. Here, a person is male if his height is more than 180 cm or if his height is less than 180 cm, he is 80 kg. otherwise, it is female. Have you ever thought about how you arrived at this decision tree? Let's use a weather dataset to explain.

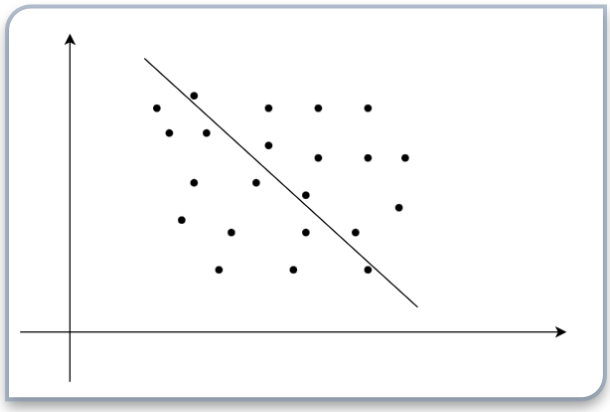

*Figure 27. With less complex data set*

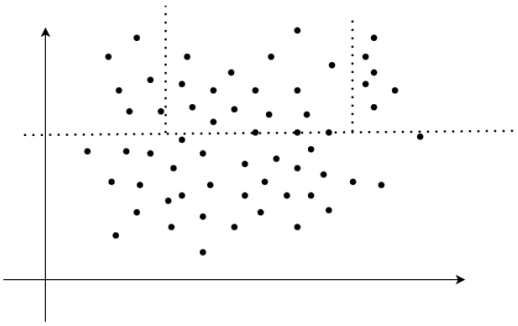

*Figure 28. With more complex data set*

## B. Entropy

In machine learning, entropy is a measure of the randomness of processed information. The higher the entropy, the more difficult it is to draw conclusions from this information.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

*Figure 29. Entropy*

## C. Information Gain

Information gain can be defined as the amount of information gathered about the signal obtained from observing one random variable or another. It can be viewed as the difference between the entropy of the parent node and the weighted average entropy of the child nodes.

$$IG(S,A) = H(S) - H(S,A)$$

Alternatively,

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

*Figure 30. Information gain*

## D. Gini impurity

Gini impurity is a measure of how often randomly selected items from a set are mislabeled when randomly labelled according to the distribution of labels within the subset [3].

$$Gini(E) = 1 - \sum_{j=1}^{c} p_j^2$$

*Figure 31. Gini impurity*

## E. Application of decision tree

- With decision tree prediction of the size of the truck that should make a move to pick the bales and number of repetitions that should take place is done.
- Data set used are number of bales to be picked, size of the bale, Size of the vehicle and repetitions.

## F. Algorithm

- Loading the data from the csv file created, which contains all required data. Import is done by pandas.
- Dividing the target variables with the independent variable. Target variable are Vehicle size and number of repetitions, independent variables are bale size and number of bales to be picked.
- Encoding the target variables, i.e., converting it into number format.
- Dropping the default columns i.e., the uncoded target column.
- Importing decision tree from sklearn
    - Sklearn (scikit-learn) is an easy-to-use framework containing a variety of useful tools such as classification, regression, clustering models, pre-processing, dimensionality reduction, scoring tools, and more [2].
- Training the model.
- After all these courses of action the model predicts the outcome.

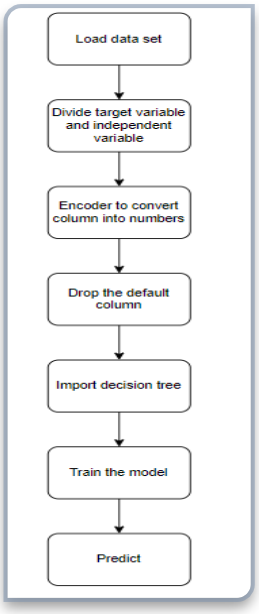


*Figure 32. Flow diagram*

## X. CONSTRAINTS AND FLOW DIAGRAM

### A. *Contraints*

| Size of the bale/ Size of truck | Big | Medium | Small | Weed |
|---|---|---|---|---|
| Big | 1 | 1 | 2 | X |
| Medium | 0 | 1 | 1 | X |
| Large | 0 | 1 | 2 | X |
| Weed | X | X | X | 1 |

*Figure 33. Constraints*

### B. *Flow diagram*

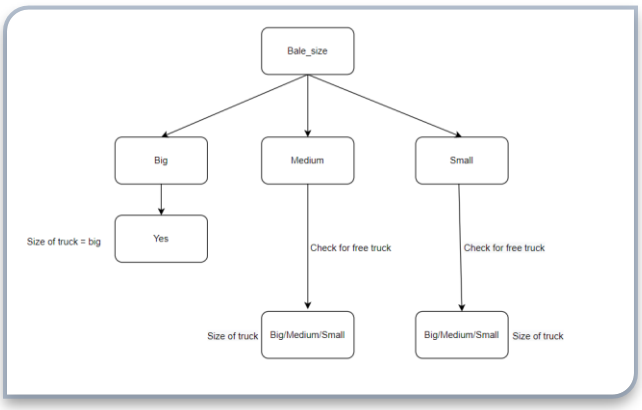


*Figure 34. Flow diagram*

## XI. RESULTS AND DISCUSSION

Decision tree methods are powerful statistical tools for classification, prediction, interpretation, and data manipulation, and have several potential applications in medical research. Using a decision tree model to describe research results has the following advantages:

- Easy to understand and easy to interpret.
- Nonparametric approach without distribution assumptions.
- Easily handle missing values without resorting to imputation.
- Easily handle highly skewed data without resorting to data transformations.
- Robust against outliers.

Like all analysis methods, the decision tree method has limitations that the user must be aware of. The main drawback is that overfitting and underfitting can occur, especially when using small datasets. This issue can limit the generalizability and robustness of the resulting model. Another potential problem is that strong correlations between various potential input variables may lead to the selection of variables that improve model statistics but are not causally related to the outcome of interest. Therefore, care should be taken when interpreting decision tree models and using the results of these models to hypothesize causality.

## REFERENCES

[1] https://scikit-learn.org/stable/modules/tree.html
[2] https://favtutor.com/blogs/gini-impurity
[3] https://uppaal.org/