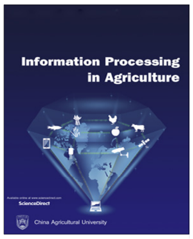
Available at www.sciencedirect.com

INFORMATION PROCESSING IN AGRICULTURE 7 (2020) 152–164

journal homepage: www.elsevier.com/locate/inpa



A hybrid weed optimized coverage path planning technique for autonomous harvesting in cashew orchards

Kalaivanan Sandamurthy [*], *Kalpana Ramanujam*

Pondicherry Engineering College, Pillaichavady, Puducherry 605 014, India

ARTICLE INFO

Article history:
Received 22 September 2018
Received in revised form
16 March 2019
Accepted 8 April 2019
Available online 10 April 2019

Keywords:
Coverage path planning
Weed optimization
Mahalanobis distance
2-Opt operator
Harvesting
Robotics

ABSTRACT

A coverage path planning algorithm is proposed for discrete harvesting in cashew orchards. The main challenge in such an orchard is the collection of fruits and nuts lying on the floor. The manual collection of fruits and nuts is both time consuming and labour intensive. The scenario begs for automated collection of fruits and nuts. There are methods developed in research for continuous crop fields, but none for discrete coverage. The problem is visualized as a graph traversal problem and paths for autonomous maneuvering are generated. A novel Mahalanobis distance based partitioning approach for performing coverage is introduced. The proposed path planner was able to achieve a mean coverage of 52.78 percentage with a deviation of 18.95 percentage between the best and worst solutions. Optimization of the generated paths is achieved through a combination of local and global search techniques. This was implemented by combining a discrete invasive weed optimization technique with an improved 2-Opt operator. A case study is formulated for the fruit picking operations in the orchards of Puducherry. The performance of the proposed algorithm is benchmarked against existing methods and also with performance metrics such as convergence rate, convergence diversity and deviation ratio. The convergence rate was observed to be 99.97 percent and 97.83 percent for a dataset with 48 and 442 nodes respectively. The deviation ratio was 0.02 percent and 2.16 percent, with a convergence diversity of 1.18 percent and 30.14 percent for datasets with 48 and 442 nodes. The achieved solutions was on par with the global best solutions achieved so far for the test datasets.

1. Introduction

India is the second largest producer of cashew in the world. During the year 1990 the production of cashew in India stood at 294,590 metric ton, which witnessed an increase of 613,000 metric ton in the year 2010 [1]. During the year 2016–2017 the export of cashew stood at 82,302 metric ton valued at 5168.78 crore rupees [2]. The area of plantation is increasing due to the crops' inherent drought resisting capacity. It is also driven by the increase in demand in both global and domestic market. The crop is labour intensive during the period of harvesting. The crop has a unique flowering pattern where the nut appears before the fruit and matures along with the fruit. Both the nut and fruit will fall off to the ground in the

* Corresponding author at: Department of Computer Science and Engineering, Pondicherry Engineering College, Pillaichavady, Puducherry 605014, India.
E-mail addresses: kalai_4390@yahoo.co.in (K. Sandamurthy), rkalpana@pec.edu (K. Ramanujam).

Peer review under responsibility of China Agricultural University.
https://doi.org/10.1016/j.inpa.2019.04.002
2214-3173

same time. Since a well grown tree can produce over thousands of nuts in a single day, individual plucking of fruits is not feasible. Fruits with nuts attached to them fall to the ground and they are collected manually each day. If the fruits are left to rest in the ground unpicked, they tend to decay quickly and damage the nut. Acute labour shortage is being witnessed in rural India due to majority of the population migrating to the cities for construction and industrial labour. The crop being a seasonal crop, produces nuts for around 4 months in the summer season. Hence it has a disadvantage in finding seasonal labour. Advanced robotics has been identified as a disruptive technology by the McKinsey institute Global Institute for its capacity to completely change the status quo [3]. Further robotics in agriculture has the potential to increase the efficiency of resources and time, while decreasing the vulnerability of workers in dangerous situations through better planning and optimization. Automobile and Manufacturing industries have already adopted robotic platforms, but only limited development has occurred in agriculture in India. As of 2018, 100 percentage of cashew harvesting in India is performed manually. The agriculture industry is also more prone to injury and fatality due to nature of terrain and presence of venomous reptiles. This was the motivating factor to develop an autonomous system for harvesting in cashew orchards. As yet, to the authors' knowledge, there exists no technique specifically developed for harvesting cashew nuts from the orchards.

2. Related works

A cherry harvesting robot was developed by Tanigaki et al. [4] to tackle the shortage of seasonal labour in Japan. The positions of cherries were calculated using the data from two pulsed-laser beams. By analyzing the data from the reflected light spectrum, cherries and obstacles were classified. Soni et al. [5] developed a tree-pruning device which required a human operator to be correctly positioned on a tree. The device used wheels to encircle and propel itself along the tree trunk. An autonomous tree climbing robot was developed by Lam and Xu [6]. The use of grippers and a flexible body enables it to climb trees and explore the environment for tasks such as animal observation or tree maintenance. A robotic arm for the purpose of harvesting radicchio was reported by Foglia and Reina [7]. A linkage system was used to pick out the vegetable from the soil. Every seven seconds one radicchio plant was harvested. A static arm was mounted to a moving tractor to perform this task. A monocular camera was used for computer vision to localize the target. The research in agriculture has focused on spraying, tilling and pruning rather than on picking.

The work on coverage path planning in agriculture has always considered applications involving heavy machineries such as tractors for field operations. Jin and Tang [8] addressed the issue of spreading seeds using tractors on farmlands. They used elevation models to identify sub regions in the terrain. Instead of adopting a single solution to the whole field, the sub regions were individually optimized according to their unique features. Parallel work paths were formed by offsetting the curve to the edges of the workspace. The task of applying fertilizers was addressed by Hameed [9]. A path planning approach extending to 3D terrain was also developed using field elevation data. An energy consumption model was implemented for planning the optimal driving angle thereby minimizing the energy needed to travel in parallel paths along the farmland. However, these approaches were designed to address the issue of complete coverage of a large farmland with tractor type of vehicle. The only objective was to maximize the total area covered. The crops considered in the works were in a highly structured environment perfectly aligned in rows or parallel paths having simple geometries. The planning of orchard operations to form a graph problem was generalized by Bochtis et al. [10]. Row wise operations such as mowing, spraying and harvesting was to be achieved in their case. The vehicle routing problems were mapped as generalized case of Travelling Salesman Problem by Bochtis and Sorensen [11]. The start and end points of each row were represented by nodes, and a matrix of connection costs were generated to form the graph. This approach was extended by Bochtis et al. [12] to derive a path for wildlife avoidance based on escape model for the animals. A machine learning approach aided with human and computer vision for planning paths in intricate environments was proposed by Bao et al. [13]. Path planning for stochastic models with a machine learning approach was proposed by Kang et al. [14]. Depth First Search (DFS) and Breadth First Search are the commonly used algorithms to traverse graphs. The DFS explores the children nodes first before moving to the level above whilst nodes on the same level are first explored before moving on to the children nodes in BFS. The main difference between these two techniques is the data structure used to store the traversal. Breadth First Search uses a queue while Depth First Search uses a stack based implementation [15]. A graph searching algorithm was presented by Fatime [16] to determine the distance between two specific nodes. This approach however required the maintenance of huge arrays that grew as the problem scaled. A nearest node approach for the Travelling Salesman Problem was implemented by Kirk [17]. The algorithm determines the route by visiting the next closest node, and proceeds until the end node is reached.

The novel contribution in this paper is solving coverage route planning problem for a node to node harvesting machine. This could have applications in a wide variety of robotics, which require visiting multiple discrete points in a search area. They find application even outside harvesting such as security, search and rescue and also sampling of points of interest (used in exploration of land and space). The performance measures such as time taken, fuel consumption, and vehicle lifetime can be improved by optimizing visitations to the points of interest. Currently, the state of the art practice for coverage path planning applied to harvesting is being limited to continuous crops such as wheat in large open fields: a lack addressed in this work. A partitioning strategy based on Mahalanobis distance approach is developed to reduce computational complexity, allowing integration of other planners that were impractical for large graphs. A case study is considered based upon the data collected from cashew orchard located in the union territory of Puducherry.

In this paper, the methodology of the proposed distance model is introduced first. An algorithmic technique based on the behavioral pattern of spreading of weed is adopted to optimize the planning of path. The proposed method is then benchmarked against existing algorithms and results are analyzed.

3. Methodology

The former case of coverage can be an ill-fitting abstraction in many applications for harvesting. A more fitting approach would be that of a node visitation problem when the object to be harvested is discrete (example cashews, radicchios) as opposed to continuous harvesting (wheat or sugarcane). A commercial cashew plantation, despite best maintenance practices, spans over a rough terrain, trees planted in rows over their 50 years of life span adopt irregular positions and poses due to various environmental factors. A single tree can cover 1000 square feet of area. Fruits attached with nuts are spread over the ground, to be picked upon manually. In this case, each element can be considered as a node, and an edge can be described as a path between the nodes. The proposed methodology considers this approach for coverage path planning specifically for the above noted purpose of discrete coverage. After harvesting a particular fruit, the nodes can be removed. Hence every node can be visited only once. After visitation a node cannot be used as a stepping stone to traverse to other nodes in the graph. Mahalanobis distance between objects to be picked was calculated based on the fruit distribution data. Initial paths were formed by means of a partitioned path planning strategy. Optimization of paths were achieved by combining a global and local search technique. The proposed technique was able to achieve optimal coverage. The overall architecture of the proposed system is depicted in Fig. 1.

3.1. *The commercial cashew orchard used in this case study*

The research was based on the cashew orchard located at 12° 02′29.8″N, 79°51′02.1″E Puducherry, India to have a reference for our model. The orchard was spread on an area of 6475 square meter and was surrounded by orchard on all the sides. The trees aging from 1 year to 70 years were present in the farm. Three trees in the age of 20–40 years were selected for observation. This was the ideal age for a cashew tree to be in their peak production cycle. The farm was located on a coastal region and was receiving ample sunshine throughout the year.

3.1.1. *Diurnal and seasonal aspects*

Since experimental data on the distribution of cashew nuts were not available, 3 trees were chosen to analyze their distribution. Flowering is induced by the seasonal changes and quality sunshine. The month from February to May is the harvesting season. The fruits and nut mature simultaneously and fall to the ground. Highest number of fruit fall is witnessed when the sun is at its peak position. Fig. 2 shows the amount of fruits that fell on the ground from a tree aged 20 years during a single picking cycle. The picking cycle is assumed to end at 18:00 h on a day. The number of fruits present in the ground at that time is zero. They were analyzed from 7 PM to 6 PM the next day. Three trees of different ages were chosen for observation. It can be inferred that irrespective of their age, maximum number of fruit fall is observed during the period between 11 AM–3 PM. During cloudy days fruits took an elongated period to mature. Hence a reduction of fruits in the ground was observed. Figs. 3 and 4 represent the number of fruits from trees aged 20 and 30 years respectively. Fig. 5 is the sample tree aged 20 years present the farm.

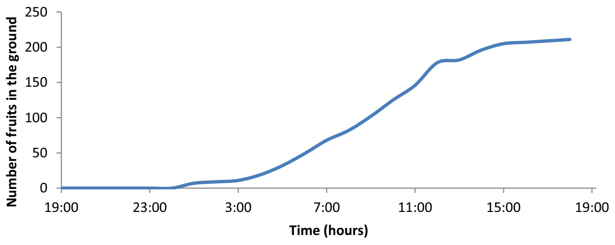


Fig. 2 – Cumulative fruit production of a tree aged 20 years in a single day, based on the fruit picking cycle.

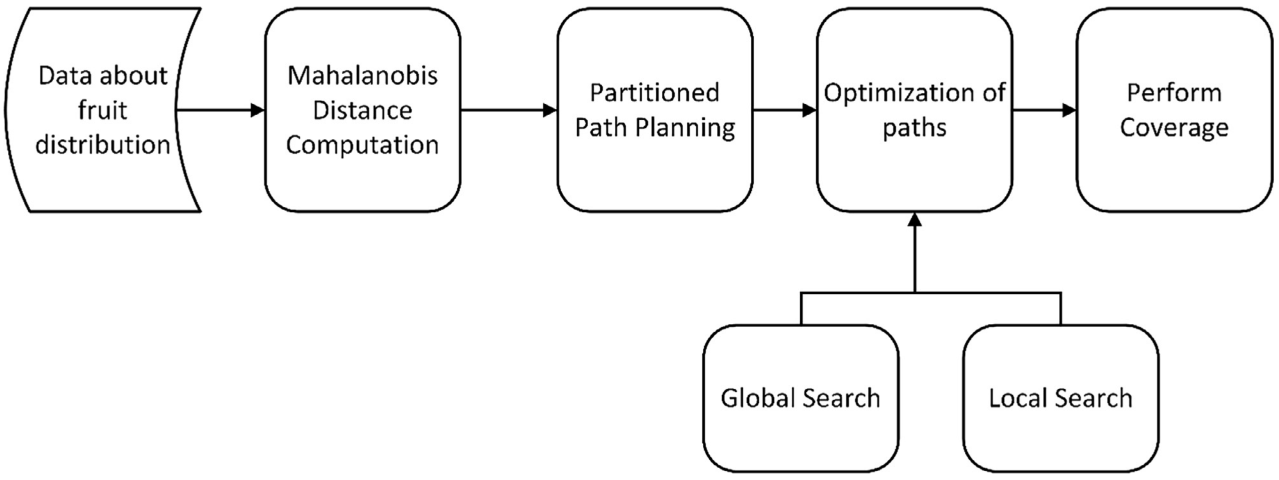

Data about fruit distribution
Mahalanobis Distance Computation
Partitioned Path Planning
Optimization of paths
Perform Coverage
Global Search
Local Search

Fig. 1 – Architecture of the proposed path planner.

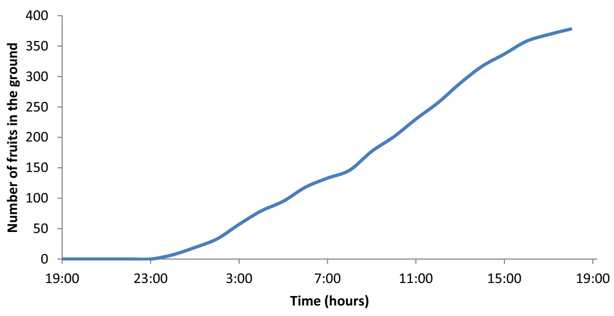


Fig. 3 – Cumulative fruit production of a tree aged 30 years in a single day, based on the fruit picking cycle.

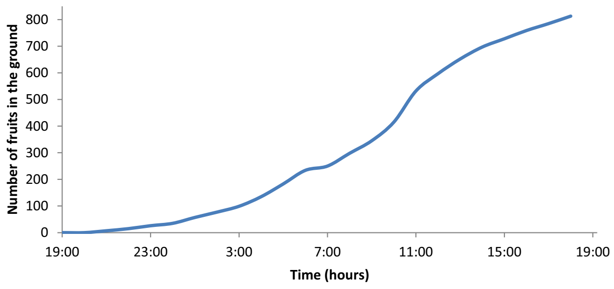


Fig. 4 – Cumulative fruit production of a tree aged 40 years in a single day, based on the fruit picking cycle.

Fig. 5 – Sample tree aged 20 years.

3.1.2. *Location specific aspects*

The harvesting season occurs when the sun transits from winter solistice towards equinox. Analyzing the wind pattern of the region, wind from the southern direction is prevalent during the harvesting season. The trees which received direct sunshine without any hindrance from other trees and branches were able to produce more fruits when compared to trees which received partial sunlight. Higher fruit production were observed in trees which received a combination of ample sunshine and wind flow around them.

4. Mahalanobis distance based partitioned path planning

In case of harvesting applications, a node can be visited only once and hence the ideal case occurs when all nodes are visited. Through permutations the maximum number of complete paths can be determined a priori. There cannot be more than n! Complete routes for a given set of n nodes. The partial paths not visiting every node are excluded. The path planning approach must account for the non-uniform distribution of fruits with respect to time (the period spent in the ground) and space (the location in a search space). The key requirements of the path planner is as follows

1. Minimize the time spent by fruits lying on the ground.
2. Collect all the fruits present in the search space.

Every fruit to be collected had two components associated with it namely, the time of fall and the location of fall. The time spent by fruits lying ideal in the ground had to be minimized for efficient harvesting. There was one more objective of covering all the fruits (nodes) with minimum distance. The conventional path planners for such graph traversal algorithms used the squared Euclidean distance based model for finding the shortest path. i.e. finding an Hamiltonian path to traverse through the graph. Since every fruit has two deciding components associated with it, Mahalanobis distance was selected to exploit the correlation between the objects' associated components. Mahalanobis distance can be defined as the distance from vector y_j to the set $X = \{x_1,\ldots,x_{nx}\}$ as the distance from the centroid of x, y_j to $\acute{x}$, weighted to the variance matrix of the set X, C_x [18]

$$d_j^2 = \left(y_j - \acute{x}\right)' C_x^{-1} \left(y_j - \acute{x}\right) \tag{1}$$

where $\acute{x} = \frac{1}{n_x}\sum_{i-1}^{n} x_i$ and $C_x = \frac{1}{n_x - 1}\sum_{i=1}^{n_x}(x_i - \acute{x})(x_i - \acute{x})'$

The formula for calculating Mahalanobis distance is given by Eq. (1). Mahalanobis distance for a given fruit distribution is calculated by combining the two parameters associated with every fruit. i.e. the distance to nearest group and also the amount of time the fruit spent in the ground. The returned number is unitless and is used to correctly classify a fruit as to in which partition set it has to be placed. The routes are stored using a matrix. It would have dimensions of (n!, n) or even more if all partial routes are stored. Thus the storage would rise exponentially when n is increased. When stored as 64-bit numbers it would be size = 8.n.n! bytes. EXAMPLE at n = 10 nodes the memory requirement is 2.8 Gigabytes. Therefore it is required to partition the dataset such that not more than τ nodes are present in any single set.

The memory requirement is reduced by developing a recursive partition strategy wherein only the subsets are planned and then the sub paths are joined with a global path to form a complete solution. Fig. 6 illustrates the partition strategy based on Mahalanobis distance. Initially there are 28 fruits. The τ value is set to 3. Since the initial number of fruits is greater than τ, the group is partitioned into τ subsets by grouping the fruits based on the calculated Mahalanobis distance. This is the first step of recursion. The partitioned subset now contains 9, 7 and 12 fruits respectively. The order of traversal through the subsets is now planned. Since there

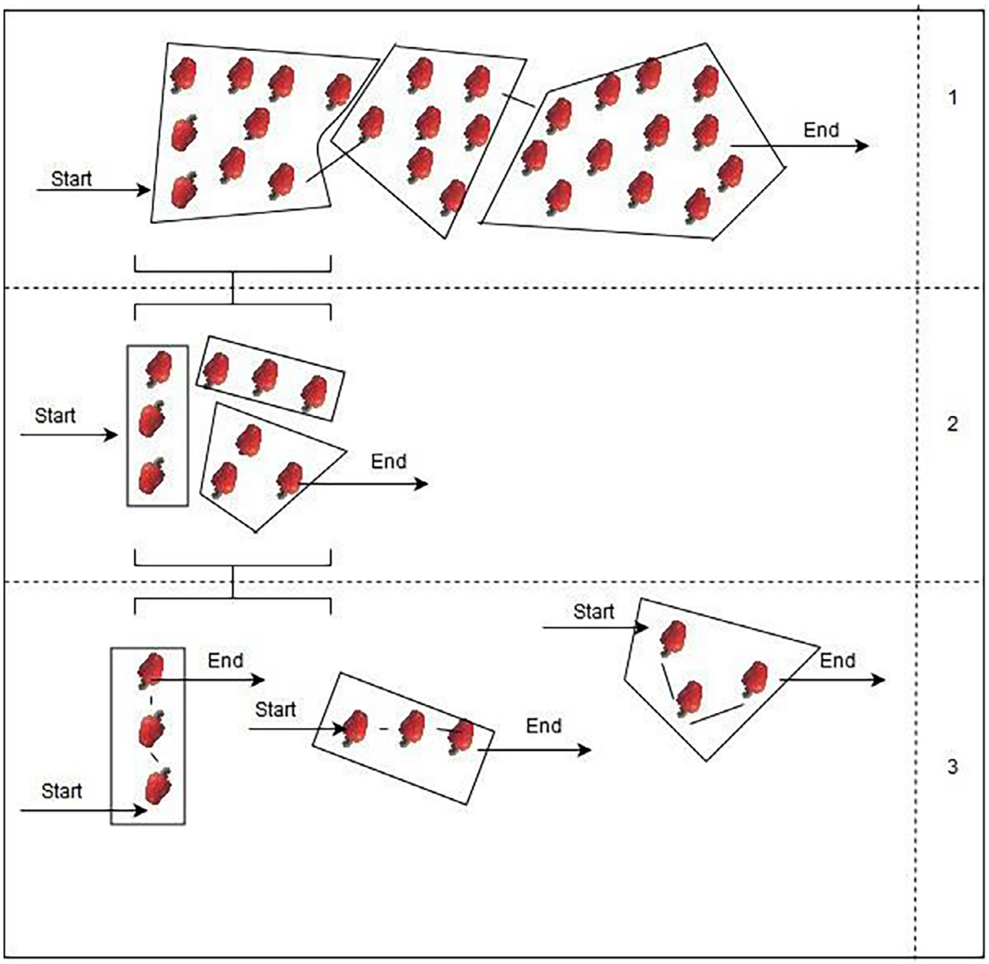


Fig. 6 – Partitioning strategy based on Mahalanobis distance for $\tau = 3$.

are more than τ fruits in each subset, the area is again partitioned to complete the path planning procedure. The subset containing 9 fruits is partitioned into three subsets defining the second level of recursion. The number of fruits in the obtained subset does not exceed the τ value. Hence a path is planned between the fruits which form the starting of a global path. The procedure returns to the first level and plans a path through the next subset in a similar fashion. The resulting path is added to the global path. The pseudo code for the partition strategy is detailed below

Pseudo code for partitioning strategy

```
Define the start point, end point
Nodes = locations to visit
If (length (nodes) > $\tau$)
   Form subsets by partitioning nodes based on
  Mahalanobis distance
   Plan a path touring through the centroids of subsets
   For each subset in path
      Recurse with nodes set to data from subset
   End
Else
   Plan path through nodes
End
```

The algorithm starts by randomly grouping each data point to one of the τ groups. The nodes are reassigned iteratively, to minimize the cost associated with grouping of nodes. The cost function is based on the calculated Mahalanobis distance metric. A local minima has been found once the cost does not change for repeated iterations and the grouping is complete. Graph connection information is made available in the adjacency matrix. It is defined as in Eq. (2).

$$\gamma_{i,j} = \begin{cases} 0, if no edge(A_i, A_j) \\ C(\theta_k), otherwise \end{cases} \tag{2}$$

An edge (θ_k) joins two nodes(A_i, A_j). Each element of the matrix would represent the cost $C(\theta_k)$ to travel along the given edge. The zero valued elements are non traversable, hence considered irrelevant in the matrix. If a matrix has in the order of n non-zero elements, it is considered sparse for large n [19]. Matlab has a sparse class to store such matrices efficiently. An example provided in [20] justifies the use of sparse matrices for such scenarios. For n = 10000, 100 million values are required to store in a standard matrix format and would occupy around 800 megabytes of memory. It would require only 0.25 megabytes of memory if stored in sparse format. Instead of using infinity as edge cost for non-existent edges, zero is to be used to achieve this memory optimization. The edges in the graph are bidirectional. Hence the adjacency matrix would be symmetrical along the diagonal ($\gamma_{i,j} = \gamma_{j,i}$). The storage size can further be optimized by utilizing only the upper triangular area of the matrix. The full matrix can be obtained by an OR operation between the upper half of the matrix and its transposition. Edges not beginning in the current set have a cost set to zero, and edges not joining the next partition set also have their cost set to zero.

In this work, the number of nodes visited has a higher priority than the cost of the route taken. The start and end nodes are defined in the planning stage. The research data indicates the starting position for our problem. i.e. the place which has highest distribution of fruits. The planning algorithm is detailed below.

Pseudo code for Path planner

```
Start point, end point        /pre-defined
Path group = start point       / Initialization of the set
of paths from the start node
For each path in path group:
  Next nodes group        /Nodes from adjacency matrix
connected to the end of path
  If (next nodes group is empty)
    Delete path from path group
  Else
    For each node in next nodes group:
    Add node to path to create the new path
  If (node == end point)
    Delete path from path group
    Save new path in final path
  Else
    Add new path to path group for next iteration
      End
    End
    End
  End
Return final path
```

The ROC Curve values for Euclidean, Manhattan and Mahalanobis distance measures are illustrated in Fig. 7. The Area Under Curve value is highest (0.8676) for classification based on Mahalanobis distance. To benchmark the proposed algorithm, existing graph traversal techniques such as Depth First Search (DFS), and Traveling Salesman Problem with nearest neighbor approach (TSP_NN) were considered. The algorithm was tested with six datasets containing 100–600 nodes. They were modeled based on the input obtained from the reference farm. Table 1 details the coverage obtained by the algorithms. Table 2 presents a statistical analysis of the results obtained from Table 1. TSP_NN implements an exhaustive search to cover the nodes. Hence it was able to attain maximum coverage but took huge time to complete the coverage.

5. Discussion

By ignoring the non-existent edge data which constitutes for more than 80 percentage of the total memory requirement, an optimization in storage space is achieved. Since it is assumed that the edges are unidirectional, the matrix becomes symmetrical along the main diagonal. The entire matrix can be obtained using one half of the matrix. It can be inferred from the results that DFS can find a reliable route by covering the nodes over a range of datasets. While traversing the DFS does not take into account the end point relative to the starting point for coverage. Hence, if the end node is identified at an early stage in the traversal, a short path is returned. Since Mahalanobis Distance based partitioned path planning (MDPP) considers both starting and end point, a path is planned such that it visits as many locations possible before reaching the end point. Also the heuristically valued edge costs defined in the adjacency matrix are not considered in the DFS implementation. The adjacency matrix is read in a binary fashion. The path cost is just the summation of edge

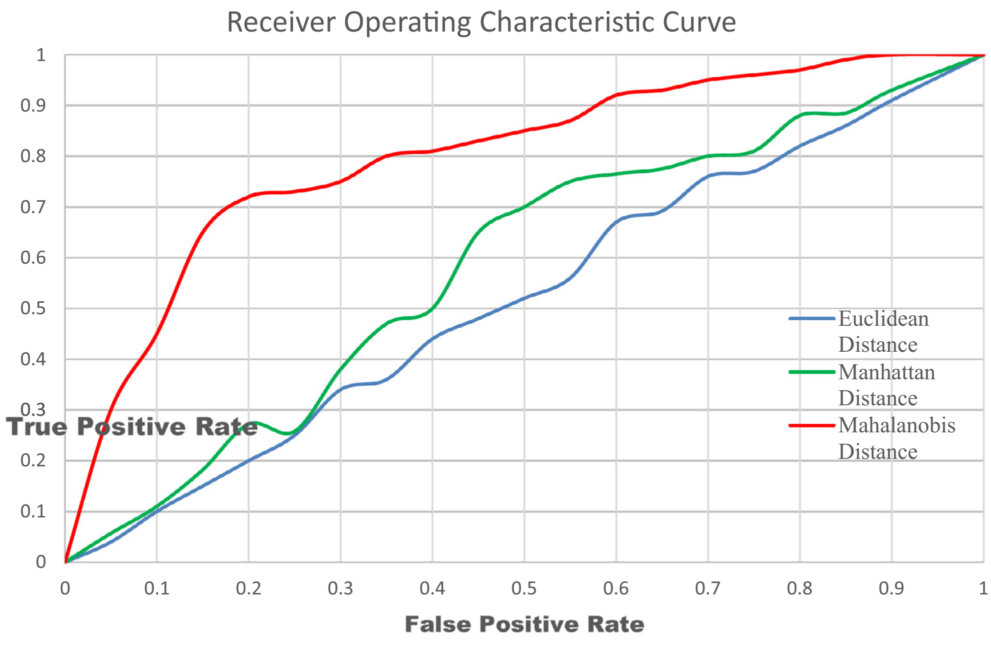


Fig. 7 – AUC curve for Euclidean, Manhattan and Mahalanobis distance.

Table 1 – Coverage obtained by various algorithms.

Number of nodes	MDPP	DFS [21]	TSP_NN [21]
100	76.00	35.00	100
200	62.00	67.50	100
300	55.00	59.33	100
400	63.40	37.25	100
500	34.00	10.80	100
600	26.29	16.00	100

Table 2 – Statistical analysis for data presented in Table 1.

	MDPP	DFS	TSP_NN
Maximum coverage (%)	76.00	67.50	100.00
Minimum coverage (%)	26.29	10.80	100.00
Mean coverage (%)	52.78	37.64	100.00
Standard deviation in coverage	18.95	22.61	0

costs obtained. There is no optimization done for the edge costs. Its single objective is to find the end node.

The objective of coverage path planner is to maximize the number of nodes visited. TSP_NN always returned the best coverage. It was found to consume huge time to find a solution, in the order of hours. Few of the paths taken were also found to be outside of the search space. The proposed MDPP achieved a mean coverage of 52.78% with standard deviation of 18.95 outperforming the next best algorithm, DFS. The proposed approach takes a holistic view by visiting as many subsets as possible before reaching the end point. The maximum group size is also limited due to the planning component associated with the algorithm. Large groups are found to increase memory space and computational time. Due to these challenges it is proposed to optimize the path planning procedure using an evolutionary algorithm – invasive weed optimization which is based on the colonizing property of weed plants.

6. Optimization

Nature-inspired algorithms are inspired by natural processes; in the recent past, many algorithms have been developed taking inspiration from certain biological, geographical, and chemical processes that take place in the natural world. Nature-inspired algorithms have the immense potential for optimizing complex real-world problems and have been applied in various areas [22]. An algorithm inspired from the colonization behavior of weed plants mimicking the vigorous, invasive habits for growth known as invasive weed optimization was proposed in [23]. Due to the inherent nature of this technique, weed optimization was chosen for optimizing the collection path.

To evaluate an optimized path for collection, datasets from TSPLIB (library of sample instances for Travelling Salesman Problem) were taken to benchmark the proposed optimization strategy. Two methods are employed for solving such problems, one exact methods and the other approximation algorithms. Exact methods can achieve optimal solution, but there is an exponential increase in time, when the problem is scaled up. Exact methods include dynamic programming [24] branch and bound [25]. Approximation algorithms can converge to a solution in polynomial time and can be further classified as local search techniques and heuristics. 2-Opt [26], 3-Opt [27], Inner over [28], LK [29], LKH [30] are few of the local search operators. They can efficiently find a local optimal solution. The heuristic methods produce near optimal solutions which is sufficient enough for a time constrained problem in hand. The few of the existing methods include genetic algorithm [31], simulated annealing [32], particle swarm optimization [33] and ant colony algorithm [34]. These algorithms have a strong global search capability and do not depend on a particular problem. More often they tend to fall into local optimum easily. To achieve the best of both worlds, a combination of local search and metaheuristics has been the recent trend in proposing hybrid algorithms. Few of them include discrete cuckoo search [35], discrete glowworm swarm optimization [36], and genetic algorithm with 2-Opt [37].

Here a discrete invasive weed optimization with a modified 2 opt technique is proposed to solve the problem. Since real time datasets for large instances were not available for cashew orchards, the problem of coverage path planning, which in literature [38] has been modeled as coverage salesman problem and evaluated with travelling salesman problem instances.

6.1. *Invasive weed optimization*

The algorithm has strong robustness, and easily fits for the problem in hand. It has also been applied in many fields such as permutation flow shop scheduling problem [39], piezoelectric actuator design [40], encoding sequences for DNA [41], PID controlled design [42], constrained optimization of combustion for coal powered boilers [43]. Set of all weeds is known as a population and weeds represent the feasible solutions of a problem. Initially a finite number of weed is dispersed over the search area. Depending on its fitness, they produce new weeds until the maximum number of weed is reached. Weeds with better fitness survive and reproduce, others are eliminated. This process continues till the maximum iteration is reached. The weed with best fitness would be closest to the optimal solution. Such algorithms produce near optimal solutions in comparably less time. They are chosen for their property of faster convergence when the problem is scaled up. Fig. 8 represents the flow of Invasive Weed Optimization technique.

Pseudo code for weed optimization

```
Initialize X Weeds with each k Memory (for maximum k
nodes)
Place X Weed at Source Event or at Random Event
Start i = 1 to Iteration_countFor Loop
Start j = 1 to N For Loop
Select Agents ∈{1, 2,. . ., X}
Perform Spatial Dispersal
Perform Reproduction
Perform Competitive Exclusion
End j For Loop
Evaluate Fitness for each if Solution is complete
Update Global Best
Replace the less fitted ones with the more fitted ones.
End i For Loop
```

The process can be described as follows:

Step 1: Initialize a population

A population of initial solutions is dispersed over the d dimensional search space with random positions. An array with range 1 to n is considered as an individual weed, where n is number of objects.

Step 2: Reproduction

The population is allowed to reproduce based on the colony's lowest and highest fitness. The formula for producing seeds is represented in Eq. (3).

$$weed_n = \frac{f - f_{min}}{f_{max} - f_{min}}(s_{max} - s_{min}) + s_{min} \quad (3)$$

where f denotes the current weed's fitness, the best and least fitness of the current population is represented by f_{max} and f_{min}. The maximum and least value of a weed is represented by s_{max} and s_{min}. This determines the number of individual seeds.

Step 3: Spatial dispersal

This step ensures randomness and adaptation in the algorithm. The seeds are randomly dispersed over the search space by normal distribution with mean equal to zero and varying variance. This ensures that seeds abode near to the parent. Standard deviation (σ) of the random function is reduced from the initial value ($\sigma_{initial}$) in every generation.

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n}(\sigma_{initial} - \sigma_{final}) + \sigma_{final} \quad (4)$$

(σ_{iter}) is the standard deviation at present generation, $iter_{max}$ is the maximum number of iterations and the nonlinear modulation index is denoted by n.

Step 4: Competitive exclusion

The number of weeds in colony reaches its maximum (p_max) after a few iterations. A mechanism for eliminating weeds with poor fitness is followed. Each weed is allowed to produce seeds as per Eq. (3). They are then spread over the search area as per Eq. (4). After which they are ranked with their parents. Now weeds with lower fitness are eliminated and allowed to reach maximum population. Weeds and seeds are ranked together and fitter individuals are allowed to replicate. The population control mechanism is applied to the seeds, realizing competitive exclusion.

6.2. *Modified 2-Opt operation*

2-Opt algorithm randomly removes two edges from a generated tour and connects a new path created. This reconnected edge is valid only if it is shorter than the replacing edge. The resulting tour is said to be two optimal. Since random selection of edges cost time, a modified 2-Opt process is detailed below [44]:

1. A tour $t = <v_1,..,v_x, v_{x+1}, \ldots, v_y, v_{y+1}, \ldots, v_n> x = 1, y = x + 2$ is selected

2. An edge of the tour $< v_x, v_{x+1} >, x \le n\text{-}2$ is selected as first edge

3. The second edge of the tour is chosen such that $< v_y, v_{y+1}>, y \le n$. if $y = n$, then $y + 1 = 1$

4. The 2-Opt procedure will be executed if $a(<v_x , v_y>) + a(<v_{x=1}, v_{y+1}>) < a(<v_x, v_{x+1}>) + a(<v_y, v_{y+1}>)$.

$<v_{x+1}, v_{y+1}>$ will be added and $<v_x, v_y>$ will be removed.

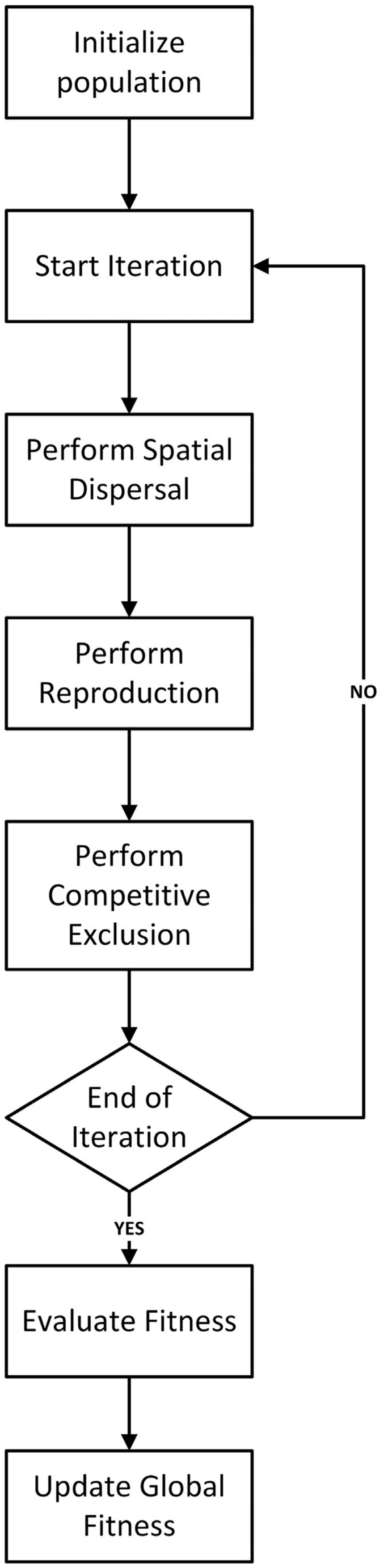


Fig. 8 – Flow of invasive weed optimization technique.

5. The path between nodes v_{x+1}, v_y, $x = 1$, $y = y + 2$ will be reversed and goto step 2

Else goto step 6

6. $y = y + 1$. If $y > n$, goto step 7 else goto step 3

7. $x = x + 1$. If $x \leq n - 2$, goto 2 else end the process

An example is shown in Figs. 9–13 detailing the 2-Opt operation. A TSP tour of seven cities are considered to explain the 2-Opt operation. From the node A, there two cross edges namely, AC and BE. They are replaced by AE and CB as shown in Fig. 10. There are two other cross edges from A, namely AE and GC which are replaced by EC and AG respectively. This is denoted in the conversion from Fig. 10 to Fig. 11. EC and BD are two other cross edges from node A, replaced by DC and EB as shown in Fig. 12. The last pair of cross edges present namely, EB and DA are replaced by ED and BA as in Fig. 13 which is the final tour without any cross edges.

Pseudo code for Weed Optimization with 2-Opt

```
Begin
Discrete Initialize (visit_weed_sequence), pop_size = N;
Iter = 1;
While iter < max_iter
visit_weed_sequence = I2Opt
(visit_weed_sequence);
For i = 1:N
Fitness (i) = F (visit_weed_sequence);
End For
Best Fitness = max (fitness);
Worst Fitness = min (fitness);
StepLength=(max_iter-iter)^n*(stepLength_ini-stepLength_final)/(max_iter)^n + stepLength_final;
For i = 1:N
Num=(max_seed – min_seed)*(fitness(i)-WorstFitness)/(Best Fitness-WorstFitness) + min_seed;
visit_sequence_seed = normrnd (0, stepLength, num);
End For
visit_weed_sequence = I2Opt (visit_weed_sequence);
If num (weed, seed) > size_pop
Pop = selectBetter (weed, seed, size_pop); N = size_pop;
Else
pop = join (weed, seed); N = num(weed, seed);
End If
End While
End
```

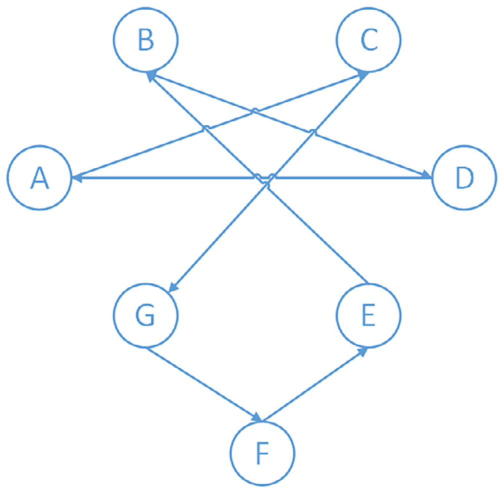


Fig. 9 – Tour ACGFEBD.

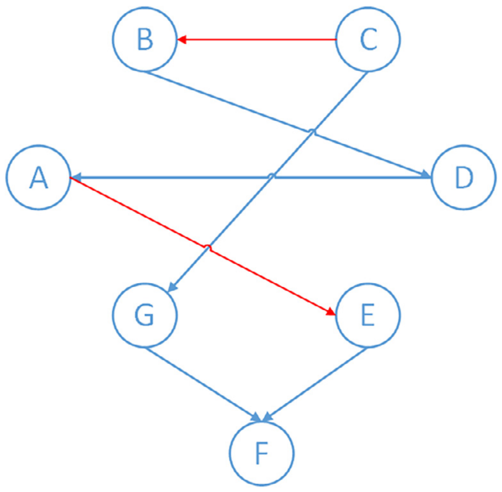


Fig. 10 – Tour AEFGCBD.

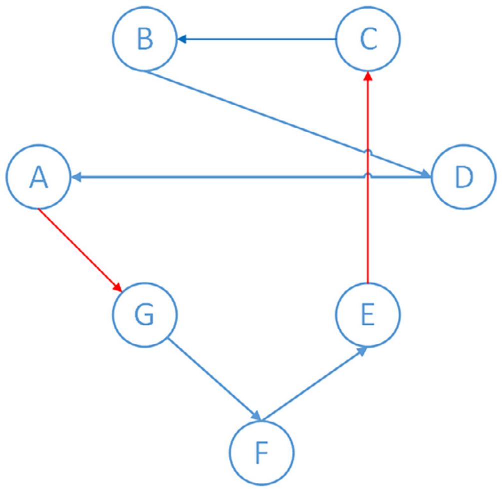


Fig. 11 – Tour AGFECBD.

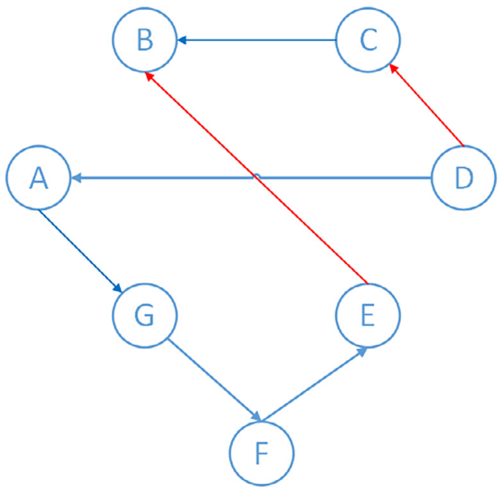


Fig. 12 – Tour AGFEBCD.

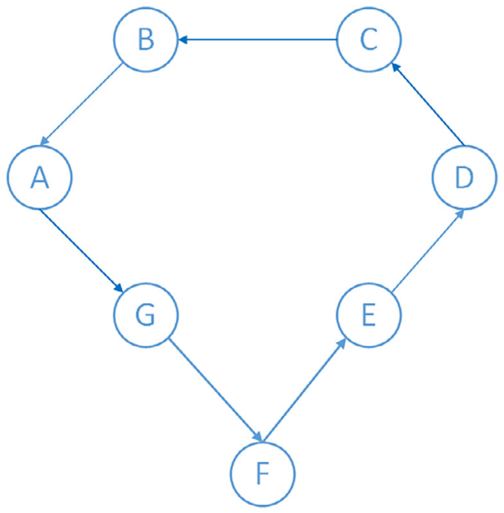


Fig. 13 – Tour AGFEDCB.

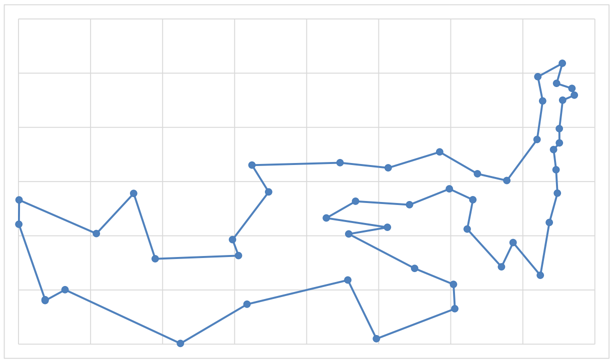


Fig. 14 – Tour of Att48.

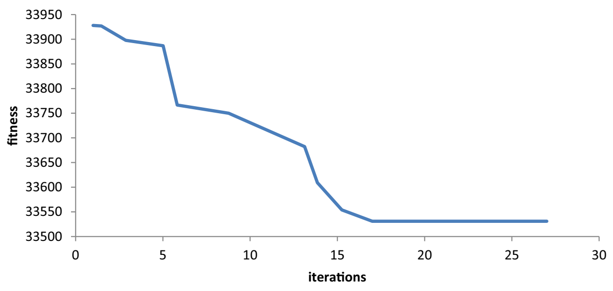


Fig. 15 – Curve evolution diagram of Att48.

Table 3 – Simulation parameter settings.

Parameter	Notation	Value
Initial population	Size_pop	10
Maximum population	Pop_max	20
Maximum iteration	Max_iter	1000
Minimum number of seed	Min_seed	1
Maximum number of seed	Max_seed	10
Nonlinear modulation index	n	4
Initial variance	Step_ini	200
Final variance	Step_final	0.1
Lower bound of individual	X_min	−1000
Upper bound of individual	X_max	1000

7. Simulation and result analysis

Experimental test cases were taken from TSPLIB. Each test case was run for 20 times and the algorithm was stopped when each run had the same test result for 5 consecutive generations.

The system settings were Processor: Intel Core i3, Frequency: 2.30 GHz, Memory: 4 GB, Software: Matlab® 2015a. The simulation parameter settings for the algorithm are given in Table 3. Each test case was run for 10 times and each run was stopped when the fitness value for 10 consecutive generations were the same.

8. Performance metrics

The performance metrics used to evaluate the proposed technique are summarized below:

1. Convergence rate: It is defined as the percentage of fitness reached by the solution with respect to known optimal solution of the problem. It is denoted by Eq. (5).

$$\text{Convergence rate } (\%) = 1 - \frac{\text{Fitness} - \text{Optimal fitness}}{\text{Optimal fitness}} X100 \tag{5}$$

Table 4 – Computational results benchmarked against 7 instances from TSPLIB.

S. No	TSP instance	Opt. value	Fitness		Conv. rate (%)		DR (%)		Conv div (%)
			Best	Worst	Best	Worst	Best	Worst	
	Att48	33,523	33,531	33,928	99.97	98.79	0.02	1.20	1.18
1	eil51	426	430	478	99.06	87.79	0.93	12.20	11.27
2	Berlin52	7542	7551	7913	99.88	95.08	0.12	4.92	4.8
3	St70	675	679.28	831.7	99.36	76.79	0.63	23.21	22.57
4	Kroa100	21,282	21,297	22,389	99.92	94.80	0.07	5.20	5.12
5	Pr107	44,303	44396.7	51826.6	99.78	83.01	0.15	16.98	16.77
6	Tsp225	3859	3927	4938.9	98.23	72.01	1.76	27.98	26.22
7	Pcb442	50,778	51876.8	67182.78	97.83	67.69	2.16	32.30	30.14

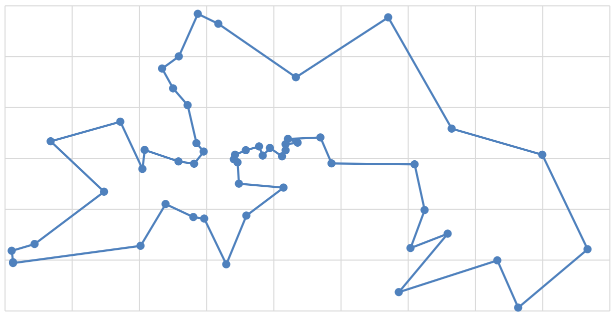


Fig. 16 – Tour of Berlin52.

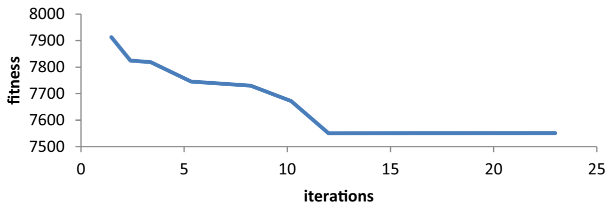


Fig. 17 – Curve evolution diagram for Berlin52.

2. Convergence diversity

It is defined as the difference between the convergence rate of best and worst solutions in the population. This factor is used to measure the diversity among individuals in a population.

Convergence diversity (%) = convergence rate$_{\text{Best individual}}$ − convergence rate$_{\text{Worst individual}}$ (6)

3. Deviation ratio is defined as the difference between fitness value obtained and the known optimal fitness for the particular instance.

$$\text{Deviation Ratio } (\%) = \frac{\text{Fitness} - \text{Optimal fitness}}{\text{Optimal fitness}} X100 \tag{7}$$

9. Result analysis

Table 4 depicts the results obtained by the proposed algorithm for 6 TSP instances. It can be inferred that close to near optimal solutions were obtained and the deviation ratio was

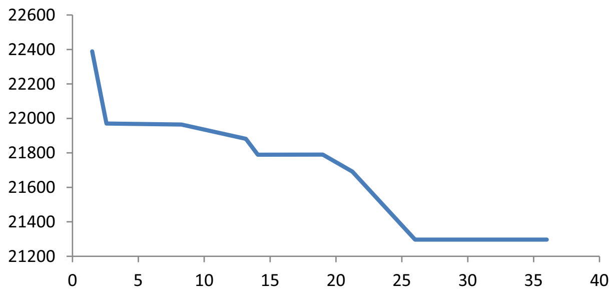


Fig. 19 – Curve evolution diagram for KroA10.

smaller. For cities less than 100, the deviation ratio was less than 1 percent. With the scale of the problem the deviation ratio was found to increase significantly. The running time was more than 45 min, even hours for large datasets, but the solution obtained has less deviation from the global optimum. Figs. 14–19 are some examples of the tour obtained and their respective curve evolution diagram for sample datasets. The convergence iteration for Berlin52 is 12, and the convergence iteration for KroA100 is around 26 which implies faster convergence of the proposed algorithm. Figs. 14, 16, 18 are the tours obtained for Att48, Belin52, KroA100 respectively. A comparison between existing techniques were made and the results are tabulated in Table 5.

10. Conclusion and future work

To generate routes for autonomous harvesting robots, an algorithm, Mahalanobis distance based partitioned path planner, was developed. A case study of fruit falling pattern in cashew orchards was analyzed. The proposed algorithm had outperformed other algorithms with a maximum coverage of 76% for a maximum of 600 nodes. Improving performance in larger datasets and refining the subset methodology are subjects for future work. Offline methodology has been used to determine the harvesting route. Intervention due to environmental factors may change the position of objects when the path is being planned and executed. Wind or rain might damage or change the position of fruits. An online implementation of route planning is to be developed to account for changes in fruit position in real time. An invasive weed opti-

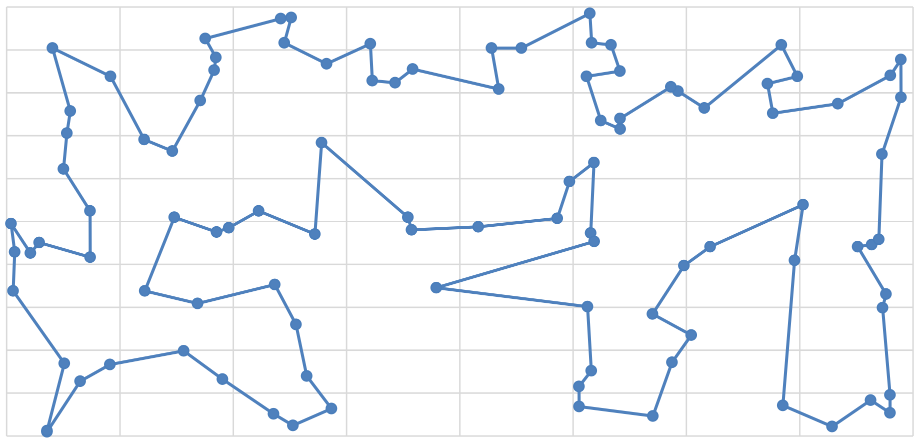


Fig. 18 – Tour of KroA1.

Table 5 – Comparison with existing algorithms.

No	Instance	Optimum	DIWO+2-Opt	DCS [35]	ACS+2-Opt [46]	SWT [45]	AGSO [36]
1	Eil51	426	430	426	431	430	428.87
2	St70	675	679.28	675	681	678.6	677.11
3	Pr107	44,303	44396.7	44,303	44,352	44,303	44,337.36
4	Tsp225	3859	3927	3916	3951	3899.9	–
5	Pcb442	50,778	51876.8	–	51788.4	52194.7	–

mization was discretized to solve Travelling salesman problem (TSP). It was benchmarked against TSP instances and results achieved were close to theoretical optimal values. The current planned paths are not able to guarantee lowest fruit lying time. The current results represent a worst case scenario where the robot collects fruits only in defined search space, whereas a farmer would collect fruits present in a nearby reachable neighborhood. An improvement in detection and collection system with use of online techniques would further enhance the performance by guaranteeing lower fruit lying time and higher coverage. A farmer cannot be collecting fruits continuously, leaving a part to be collected later. Such intervals will benefit the robot in comparison with the farmer. A next step in this research would be to develop a working model of a fruit picker. The sensitivity of various parameters such as robot speed, turning radius, energy consumption can be investigated. The presented results are promising for using a robot to collect fruits in an orchard, resulting in reduction of employing manual labor.

Conflict of interest

The authors declare that there is no conflict of interest.

Acknowledgments

The work was supported by University Grants Commission, India under the scheme NFOBC with grant no. F./201718/NFO201718OBCPON51035. The authors would like to thank the people of Namalavar Cashew Farmers Association, Puducherry for their inputs.

REFERENCES

[1] Senthil A, Magesh P. Analysis of Cashewnut production in India. Asia Pacific J Market Manage Rev 2013;2(3):2–3.
[2] Cashew Export Council of India (CEPCI). Ministry of Commerce and Industry. Government of India; 2018. Link: <http://cashewindia.org/statistics>.
[3] Manyika J, Chui M, Bughin J, Dobbs R, Bisson P, Marrs A. Disruptive technologies: Advances that will transform life, business, and the global economy. McKinsey Global Institute; 2013.
[4] Tanigaki K, Fujiura T, Akase A, Imagawa J. Cherry- harvesting robot. Comput. Electron. Agric. 2008;63(1):65–72.
[5] Soni DP, Ranjana M, Gokul NA, Swaminathan S, Binoy, Autonomous arecanut tree climbing and pruning robot. In: Proc. International Conference on Emerging Trends in Robotics and Communication Technologies (IN-TERACT) Chennai, India; 2010. p 278–282.
[6] Lam TL, Xu Y, Tree climbing robot: design, kinematics and motion planning. Springer-Verlag Berlin Heidelberg. Springer; 2012. p 23–54.
[7] Foglia MM, Reina G. Agricultural robot for radicchio harvesting. J Field Rob 2006;23(6–7):363–77.
[8] Jin J, Tang L. Coverage path planning on three-dimensional terrain for arable farming. J Field Rob 2011;28(3):424–40.
[9] Hameed IA. Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. J Intell Rob Syst 2014;74(3–4): 965–83.
[10] Bochtis DD, Griepentrog HW, Vougioukas S, Busato P, Berruto R, Zhou K. Route planning for orchard operations. Comput Electron Agric 2015;113:51–60.
[11] Bochtis DD, Sørensen CG. The vehicle routing problem in field logistics part I. Biosyst Eng 2009;104(4):447–57.
[12] Bochtis DD, Sørensen CG, Green O, Hameed IA, Berruto R. Design of a wildlife avoidance planning system for autonomous harvesting operations. Int J Adv Rob Syst 2014;11(6):6–15.
[13] Bao J, Tang H, Song A. Combining vision learning and interaction for mobile robot path planning. Int J Adv Rob Syst 2012;9(102):102–9.
[14] Kang B-Y, Xu M, Lee J, Kim D-W. ROBIL: robot path planning based on PBIL algorithm. Int J Adv Rob Syst 2014;11(147):1–14.
[15] Corneil DG, Krueger RM. A unified view of graph searching. SIAM J Discrete Math 2008;22(4):1259–76.
[16] Matlab, More efficient code to find all paths with smaller than specified length between two nodes in sparse graph; 2018. Link: <http://stackoverflow.com/questions/18169680/matlab-more-efficient-code-to-find-all-paths-with-smaller-than-specified-length>.
[17] Kirk J. Traveling Salesman Problem – Nearest Neighbor. 1.3 ed: MathWorks; 2008.
[18] Acklam Peter J. MATLAB array manipulation tips and tricks. Link <http://www.math.uio.no/~jacklam/2018>.
[19] Tewarson RP. Sparse matrices. New York: Academic Press; 1973.
[20] Lim K, Kevin L, Paul T, Michael W, Benoît S-O. LiDAR remote sensing of forest structure. Prog. Phys. Geogr. 2003;27 (1):88–106.
[21] Gleich D. Gaimc: Graph algorithms in matlab code. 1st ed. MathWorks; 2009.
[22] Shukla Anupam, Tiwari Ritu. Discrete problems in nature inspired algorithms. CRC press; 2013. p. 231–2.
[23] Mehrabian AR, Lucas C. A novel numerical optimization algorithm inspired from weed colonization. Ecol Inf 2006;1 (4):355–66.
[24] Bellman RE, Dreyfus SE. Applied dynamic programming. Princeton, New Jersey: Princeton University Press; 1962. p. 114–35.
[25] Lawler EL, Wood DE. Branch and bound methods: a survey. Oper Res 1996;14(4):699–719.
[26] Croes GA. A method for solving traveling salesman problem. Oper Res 1958;6(6):791–812.
[27] Lin S. Computer solutions of the traveling salesman problem. Bell Syst Tech J 1965;44(10):2245–69.

[28] Tao Guo, Michalewicz Z. Inver-over operator for the TSP. In: Proc of International Conference on Parallel Problem Solving from Nature, London; 1998, p. 803–12.

[29] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. Oper Res 1973;21(2):498–516.

[30] Helsgaun K. An effective implementation of the Lin-Lemighan traveling salesman heuristic. Eur J Oper Res 2000;126(1):106–30.

[31] Xiaobing HU, Shufan WU, Jiang JU. An advanced genetic algorithm for TSP. Comput Technol Automat 2000;19:34–8.

[32] Kirk Patrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.

[33] James K, Eberhart RC. A discrete binary version of the particle swarm algorithm. In: Proc of the IEEE conference on systems, man, and cybernetics, Orlando, USA; 1997. p. 4104–9.

[34] Ji Junzhong, Huang Z, Liu Chunnian. An ant colony algorithm based on multiple grain representation for the traveling salesman problems. J Comput Res Dev 2010;47:434–44.

[35] Ouaarab A, Ahiod B, Yang XS. Discrete version of the cuckoo search algorithm for the traveling salesman problem. Neural Comput Appl 2014;24(7–8):1659–69.

[36] Zhou Yong-Quan, Huang Z-X, Liu H-X. Discrete glowworm swarm optimization algorithm for TSP problem. ACTA Electronica Sinica (in Chinese) 2012;40:1164–70.

[37] Samanlioglu F, Ferrell JWG, Kurz ME. A memetic random key genetic algorithm for a symmetric multi-objective traveling salesman problem. IEEE Trans Evol Comput 2016;5:613–22.

[38] Galceran E, Carreras M. A survey on coverage path planning for robotics. Rob Auton Syst 2013;61(12):1258–76.

[39] Chen Huan, Zhou Yong-Quan, He Sucai, Ouyang Xinxin, Guo. Invasive weed optimization algorithm for solving permutation flow-shop scheduling problem. J Comput Theor Nanosci 2013;10(3):708–13.

[40] Mehrabian AR, Yousefi-Koma A. A novel technique for optimal placement of piezoelectric actuators on smart structure. J Franklin Inst 2011;348(1):12–23.

[41] Zhang X, Wang Y, Cui G, Niu Y, Xu J. Application of a novel IWO to the design of encoding sequences for DNA computing. Comput Math Appl 2009;57(11–12):2001–8.

[42] Kundu D, Suresh K, Ghosh S, Das S. Designing fractional-order PID controller using a modified invasive weed optimization algorithm. In: Proc. 2009 World Congress on Nature & Biologically Inspired Computing, Coimbatore, India; 2009. p. 1315–1320.

[43] Zhao H, Wang P-H, Peng X, Qian J, Wang Q. Constrained optimization of combustion at a coal-fired utility boiler using hybrid particle swarm optimization with invasive weed. In: Proc. 2009 International Conference on Energy and Environment Technology (ICEET), Washington, DC, IEEE Computer Society; 2009. p. 564–7.

[44] Zhou Y, Luo Q, Chen H, He A, Wu J. A discrete invasive weed optimization algorithm for solving traveling salesman problem. Neurocomputing 2015;151(3):1227–36.

[45] Siqueira Paulo, Steiner Maria Teresinha Arns, Scheer Sérgio. Recurrent neural networks with the soft 'winner takes all' principle applied to the traveling salesman problem, traveling salesman problem. Donald Davendra, IntechOpen; 2010. Link: <https://www.intechopen.com/books/traveling-salesman-problem-theory-and-applications/recurrent-neural-networks-with-the-soft-winner-takes-all-principle-applied-to-the-traveling-salesman>.

[46] Le Louarn FX, Gendreau M, Potvin JY. GENI ants for the traveling salesman problem. Ann Oper Res 2004;131(1–4):187–201.