# Vehicle Path Planning for Complete Field Coverage Using Genetic Algorithms

A.E.F. Ryerson and Q. Zhang

Dept. of Agricultural & Biosystems Engineering, Univ. of Illinois, Urbana, IL 61801, USA.

E-mail: RyersonAnneE@JohnDeere.com

## ABSTRACT

In farming operations, one of the fundamental issues facing a farmer is the cost of running the farm. If the equipment the farmer is using can be made more efficient, the cost of farming will be reduced. One way of making agricultural equipment more efficient is to develop automated or autonomous functions for the equipment. One of the fundamental tasks for autonomous equipment is to plan the path for the equipment to travel. This paper reports the research on the feasibility of creating an automated method of path planning for autonomous agricultural equipment. Genetic algorithms were chosen to plan the paths with a primary goal of creating an optimal path guiding the equipment to completely cover a field while avoiding all known obstacles. Two example fields were designed for evaluating the feasibility of this concept on simple problems. While simulation results verified the feasibility of this conceptual path planning method, they also indicated that further development would be required before the algorithm could actually be implemented on agricultural equipment for real-world field applications.

**Keywords:** Autonomous equipment, genetic algorithm, off-road vehicle, path planning

## 1. INTRODUCTION

As family farms have grown in size, often giving way to the larger corporate farm operations, many advances in efficiency have developed. Agricultural vehicles have continually grown to meet the ever-increasing demands of farming customers. In order to continue to improve efficiency and profitability, customers have demanded more horsepower, larger capacity, and comfort/convenience items, all allowing the vehicle to work for longer periods. Nevertheless, the ability to continue making improvements in areas such as horsepower and machine capacity is not infinite. Automation of agricultural vehicles has the potential to make farming more efficient (Palmer, 1991). Any automated machine needs a path on which to travel. There are two basic types of paths for automated vehicles. “Off-board” paths are often generated offline, either automatically by a computer or manually by a user. These off-board paths generally can take into account everything known in the operating environment, such as permanent obstacles and boundaries. They do not, however, have the ability to respond to unknown objects. On the other hand, “on-board” paths are those generated by the vehicle controller “on-the-run”. On-board paths generally are good at reacting to obstacles or unsafe conditions, handling unexpected situations.

The problem of completely covering an area applies to many of agricultural operations, such as harvesting, mowing, planting, and spraying. The focus of this project is to create an optimal coverage path through a two-dimensional area that could be applied to agricultural applications. Genetic algorithms, combining the natural process of “survival of the fittest” with the power and speed of today’s computer algorithms to rapidly find optimized solutions (Goldberg, 1989), have

been studied extensively in vehicle path planning. Many researchers have studied using genetic algorithms to generate routes for avoiding obstacles between a “start” and an “end” point. A common approach in attempts was to find the lowest-cost path between “start” and “end” points, and the cost was often defined as a function of the total length of the path. Some other factors, such as time and obstacle avoidance, are also considered as a factor for cost determination in some of the reported approaches (Kambhampati and Davis, 1986; Lin *et al.*, 1994; Noguchi and Terao, 1997; Sugihara, 1997; Hocaoglu and Sanderson, 1998; Gerke, 1999; Capozzi and Vagners, 2002).

The primary goal of this research was to perform a feasibility study on determining if a genetic algorithm approach is an appropriate method for performing dynamic path planning tasks and to develop recommendations for future development if it is a feasible approach. This research focused on evaluating the effectiveness of different components of a standard GA in searching for optimal paths. This research was, therefore, designed to explore a new application for GAs, rather than to create a new GA. To support this goal, several specific objectives were identified. Those research objectives are to investigate the capability of a GA in handling arbitrary obstacle shapes, to study the effectiveness of the GA-based planner on avoiding polygonal-shaped obstacles, and to lay the groundwork for future algorithm development.

## 2. ALGORITHM DESIGN

A genetic algorithm (GA) is a framework for a problem solution, not a solution in itself. Therefore, a GA must be adapted for different problems. This section describes how the GA-based path-planning problem is customized by making up of five key components to represent the field, the obstacles, the grid of travelable area, the automated vehicle, and the path.

### 2.1 Representation of Key Components

*Field Representation:* The field is the most important of all the components of this GA because it sets the groundwork for all other components. In this research, a polygonal area was chosen to represent the field of operation. This polygonal area denoted the boundaries of the field in which the automated vehicle will travel. Having an open number of vertices allowed GA to approximate, within the desired small degree of error, any possible area. To represent a polygonal field using a sequence of vertices, it requires at least three of these points to completely cover the area.

*Obstacle Representation:* One of the most important tasks in this project is the representation of obstacles in the field. In terms of the project goal of feasibility study, this project investigated two methods to model obstacles for demonstrating the feasibility. The first method was to model an obstacle as a circle, using a center point and a radius to represent its location and size. The other method was to model an obstacle as a polygon which represents an obstacle using a list of vertices (Lin et al., 1994). The detailed modeling methods will be introduced in later sections.

*Grid Representation:* Once the field and obstacles were defined, the travelable area needed to be converted into a grid of equally spaced points throughout the area. The grid spacing was defined to be equal to the machine width, and covers all the “travelable area” but not the obstacles. With the grids, the “to-be-covered” area in a field was divided into squares, and a unique index was assigned to identify the center point in the grid. The rule of assigning the index started with the

lower left-hand corner of the grid as index 0 to ensure that the grid numbering was consistent in every case, and that grid "0" was always started at the same place in this algorithm.

*Vehicle Representation:* The representation of an automated vehicle in this algorithm is essential but very simple. An assumption was made that this vehicle had an on-board controller to guide it following a provided path. The only vehicle-specific parameter used in this algorithm was vehicle width which was used to create the grids to represent the path in the field. If the size of a grid were equal to the width of the vehicle, then the machine could pass it, and the area could be considered as being completely covered during the operation.

*Path Representation:* A path in this algorithm was presented as a list of ordered 2D points for performing the functions of a genetic algorithm. A group of these 2D points served as the population in the GA, with each member being assigned a fitness value. This fitness value described the strength of individuals, and was used to determine which individuals were more likely to pass on to future generations. In this research, an ideal path was considered as the one that satisfied the following conditions that it covered 100% of the area ("the coverage measure"); it did not intersect any obstacles ("the obstacle measure"), it did not cross over itself ("the intersection measure"), and it was as short as possible ("the length measure"). The coverage measure was calculated as the total percentage of the area covered by the selected path. The obstacle measure was a count of the total number of times a path intersected an obstacle. The intersection measure was a count of the total number of times the path crosses itself. And the length measure was the length of the selected path from the start point to the end point.

*Weighting Factors:* The above four parameters were used to define a successful path. However, they do not all hold the same level of importance when determining which path is the best. To solve this problem, each of the parameters was assigned a weighting factor to quantify the impact of those parameters on the fitness value of a path and such fitness values are directly related to the coverage, and inversely related to the obstacle intersections and the length. By tuning weighting factors, the impact of the evaluated parameters can be determined.

### 2.2 Basic Functions

This GA-based path planning model includes four basic functions of selection, crossover, mutation and reproduction. Selection function is one of the basic functions in a GA system and a weighted roulette-wheel selection approach was used in this model. By this approach, the selection process was designed based on such a rule that the individuals with the highest fitness have a higher probability of being selected. This process simulates a weighted roulette wheel where each individual's slot on the wheel is sized in proportion to its fitness (Goldberg, 1989).

The crossover is a function simulating the mating of individuals from the population which takes two individuals from the population (the "parents") to create two "children" from the information in parent individuals. The simplest crossover (Goldberg, 1989) is a basic exchange of information between the two parents. A random point is selected in the parents. The function copies the information before this point from Parent 1 to Child 1, the information in Parent 2 after this point is then copied into Child 1, and the reverse occurs for Child 2. Therefore, each child carries a portion of the information from both parents.

The reproduction function performs the process of creating in a population of individuals in offspring generations (in this case, new paths). The process works in such a way that for a given

population of individuals, the function selects individuals based on their fitness, and fill a population of new individuals.

A three-action mutation algorithm was designed to mutate a path as following in this research. The three actions were deleting-a-point, inserting-a-point, and modifying-a-point. One or more actions could be implemented during a particular mutation process for creating new paths. The choice of which of the three actions to implement is random. In this implementation, a random number was applied to determine which of the mutation actions would be implemented. If the action of deleting-a-point is applied, the mutation function will delete a random point in a path to form a new one. Based on this approach, once the delete action is called, the value at a randomly selected location in the path sequence will be deleted which resulted in the new path sequence one point shorter. If the action of inserting-a-point action is applied, the mutation function will insert a new point at a random location in the path sequence, and resulted in a one-point longer new path sequence. If the modifying-a-point action is applied, the mutation function modifies an existing point at a randomly selected location in a path sequence being processed. In this research, the modifying a point mutation function was so designed that it would randomly select a point in the area grid that is not already in the existing path sequence. The selected grid point would be then used to replace an existing point in the path sequence at the selected location.

Other than the four most commonly used functions described above, the GA-based path planner created in this research consisted of three more special functions of swap, record and remove crossing to enhance the performance of these functions. The swap function was used to reorganize the points within a path sequence without completely changing the order or the number of points in the sequence. The swap function was created based on a similar function developed by Lin *et al.* (1994) which created a new path sequence by swapping a back portion of an existing sequence at a randomly selected point to the beginning of the sequence. The purpose of the swap function is to create a better path while preserving some of the genetic information by retaining the order of the points. The reorder function is used to reorganize the points in a path sequence by completely changing the order of the points in the sequence. This function was developed based on the work of Capozzi and Vagners (2002). The function took a known path sequence and reordered the points in that sequence in a completely random order. The remove crossings function is used to remove crossings or places where a path crosses itself, in a path sequence.

One very important and practical function of this GA path planner is to find a location where a path sequence crosses itself and then reorders the points in that area to remove the crossing. This function is unique in that it retains the information and the path that has been generated by the algorithm, but it reverses the direction of travel on that path and swaps two points. The first step in the function was to locate a crossing. The function did this by checking each of the segments of the path against each of the other segments in the path sequence to see if they were intersected. If the intersection point $(x_I, y_I)$ of these two lines were within the boundary of a grid comprising the X and Y coordinates of the two segments, then those two segments would be classified as intersected. Otherwise the segments were considered not intersected, and the process would continue on to assess the next pair of segments. Once an intersection was found, the next step was to remove the intersection. This was simply done by re-ordering the points around the intersection, to remove the cross. This is best demonstrated using a graphical example. As a path illustrated in figure 1, defined by sequence ABCDEFG, it can simply be

reordered to remove the intersection by reversing the order of the points between A and G. The new path is defined as sequence AFEDCBG, and shown in figure 2. After a crossing in a path has been removed, the process will continue through the rest of the path to make sure there are no other crossings. In case it was not possible to remove a crossing without creating a new one, the algorithm would stop after running a finite number of times through the path to avoid an endless loop. After a path was modified, a recalcFitness variable would be set to be "True" to signal to the system that the path was changed and should have its fitness value recalculated.

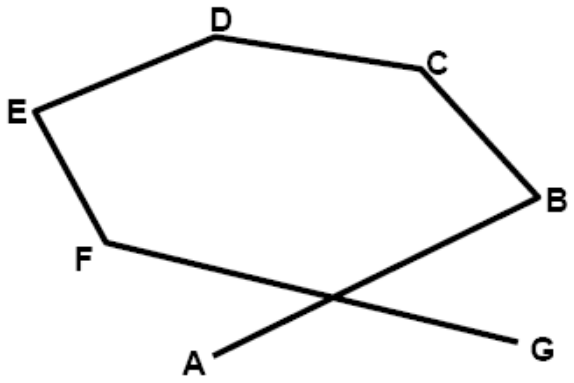


Figure 1. Seven Point Path with Intersection (before removal).

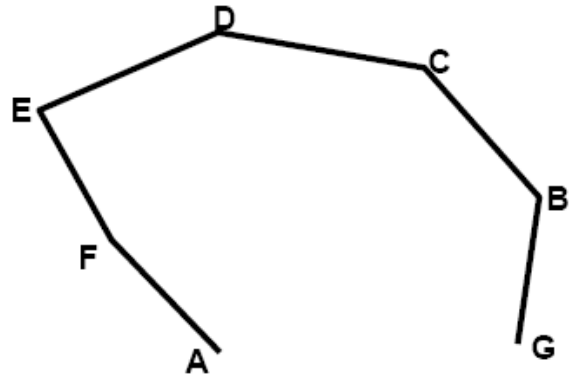


Figure 2. Seven Point Path without Intersection (after removal).

### 2.3 Integration of Basic Functions

Figure 3 shows a flowchart describing how the basic functions are integrated into the program and how the algorithm works. This research used specialized functions, in addition to the standard genetic algorithm functions, to build the path planner. This approach helps to make the genetic algorithm more problem-specific, by tuning GA functions to specific requirements of the problem. The specialized functions used in this algorithm include the swap and reorder functions, which can create new members in the population while retaining all the points in the original paths. These functions were designed based on the theory that a path might contain all the "correct" points but not in the correct order. By reordering points in a path, the algorithm could find a better path. This concept of reordering was based on the work of Capozzi and Vagners (2002). The swap function was based on the work of Lin *et al.* (1994).

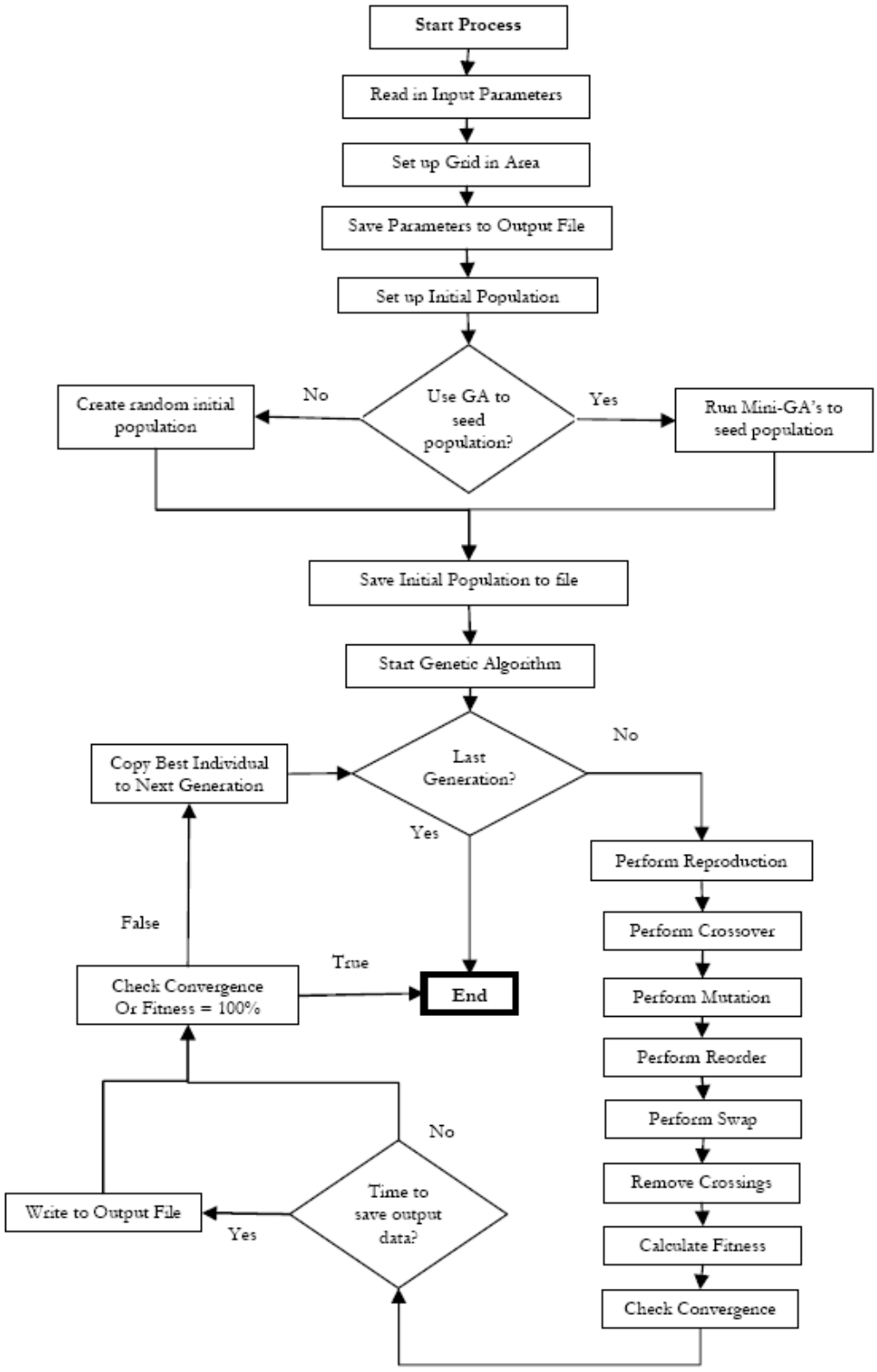


Figure 3. Flowchart of genetic algorithm (GA) based path planner.

Another function was the remove crossings function. This function allowed the paths to be "corrected" and had those portions of a path that crossed itself to be reordered in order to remove the crossing. This function searches new paths, by retaining the path information and reversing the travel direction on that path. This specially tuned genetic algorithm is able to generate paths through the field meeting the requirements of the application.

## 3. RESULTS AND DISCUSSION

A validation study was conducted based on a test condition of a field with one obstacle and a rectangular area as shown in figure 4. The purpose of the test is to evaluate the algorithm's ability to work on a simple problem and to consistently generate a path with coverage over 90% on this problem.

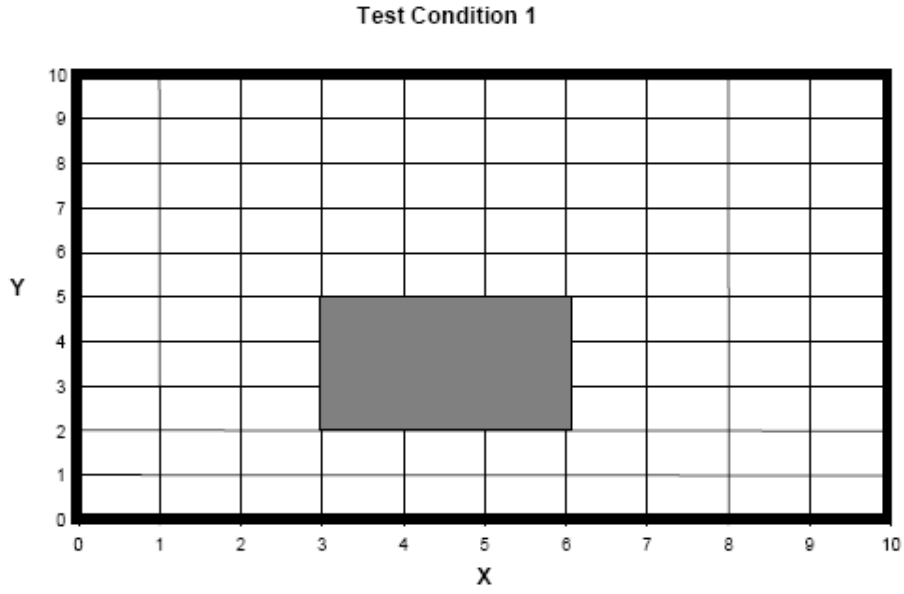


Figure 4. Area and obstacles for test condition 1.

The test settings used for this configuration are in table 1. The settings for the genetic algorithm were the highest-performing settings of 27 different groups of settings. These settings were selected after a series of 135 test runs (27 groups, 5 runs each).

Table 1. Optimized settings for test condition 1.

| Probability | Optimum Value |
|---|---|
| Mutation Probability | 0.05 |
| Crossover Probability | 0.1 |
| Swap Probability | 0.01 |
| Reorder Probability | 0.01 |

A total eight runs were performed on this test configuration. The results from those runs are shown in table 2, which shows the "best" path, or the path with the highest fitness value. A quick

review of this data showed that on four of the eight runs, the total coverage attained was greater than 90%. The average coverage attained was 89%. The top-performer was Run 5, which had a coverage of 97% and a fitness value of 30.95. This path is shown in figure 5. The lowest-performer was Run 7, which had a coverage of 82% and a fitness value of 26.37. The best paths generated by the top runs from this configuration are listed in table 3.

Table 2. Results of test configuration 1.

| Run | Coverage | Fitness |
|---|---|---|
| 1 | 0.90 | 28.84 |
| 2 | 0.87 | 27.78 |
| 3 | 0.89 | 28.48 |
| 4 | 0.86 | 27.43 |
| 5 | 0.97 | 30.95 |
| 6 | 0.91 | 29.19 |
| 7 | 0.82 | 26.37 |
| 8 | 0.92 | 29.54 |
| Average | 0.89 | 28.57 |

Table 3. Top four results from test configuration 1.

| Run | Coverage | Fitness | Figure |
|---|---|---|---|
| 1 | 0.90 | 28.84 | 6.4 |
| 5 | 0.97 | 30.95 | 6.5 |
| 6 | 0.91 | 29.19 | 6.6 |
| 8 | 0.92 | 29.54 | 6.7 |

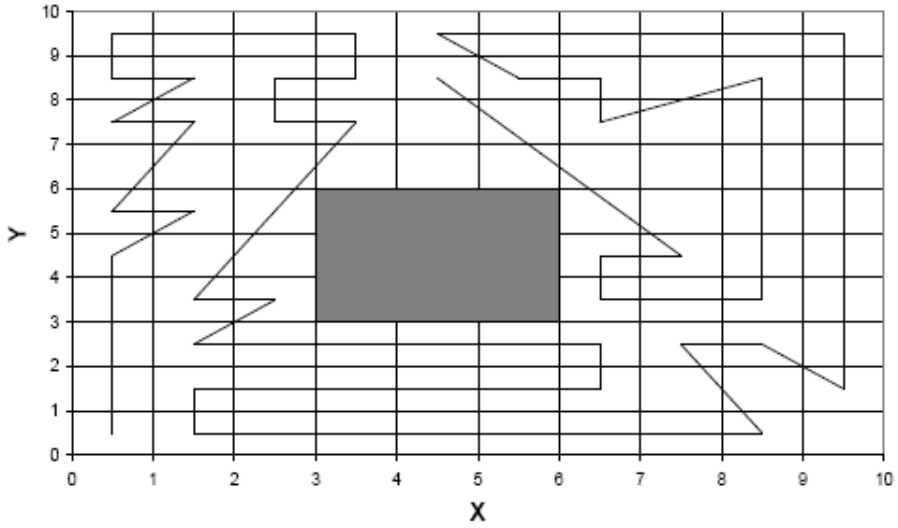


Figure 5. Results of run 5 for test configuration 1.

When comparing the results of the top four runs of table 3, there are a few interesting things to be noticed. First, the path of Run 5 had many sharp edges and turns, which would make it very difficult for a tractor to physically follow. This was because this genetic algorithm could not take turning radius of the machine into account. Such a result implies that a fully developed GA-based path planner for agricultural vehicles must be capable of modifying the path to be "vehicle-specific" in order to be able to physically follow it. Next, in order to attain a higher coverage, the algorithm had a mutation function that added points to the path. Therefore, adding these points caused the path to become more jagged. This means that the algorithm needs to add more points to this path in order to get a higher coverage, however it may result in sharp edges observed in the final path. Last, it is important to remember that a genetic algorithm is still a random process. Each time the genetic algorithm runs, it starts with a completely new random population. Therefore, the results of each run of the genetic algorithm will result in a different solution, and these four solutions are intrinsically different due to the random nature of the genetic algorithm. This is one of the basic characteristics of a genetic algorithm.

Other observations made on the final paths generated by the genetic algorithm indicated that the genetic algorithm could successful generate paths not intersecting any obstacles. The route a resulting path created remained at least a half-grid away from the obstacle. Since the grid was equivalent to the width of the machine, it could be concluded that the path could successfully avoid the obstacle. Another observation was that the resulting paths were not the "typical" paths as a farmer would traditionally take. That was because the genetic algorithm used in this planner did not have features built in for "encouraging" parallel paths. By adding such a consideration in future GA planner development, it is possible to add such functions into the GA system, and generate paths more like a human operator would normally take.

Similar validation studies were conducted on different field configurations, such as on a non-rectangular field with multiple obstacles. Very similar results were obtained from those studies. Figure 6 shows the results obtained by running these groups of settings.

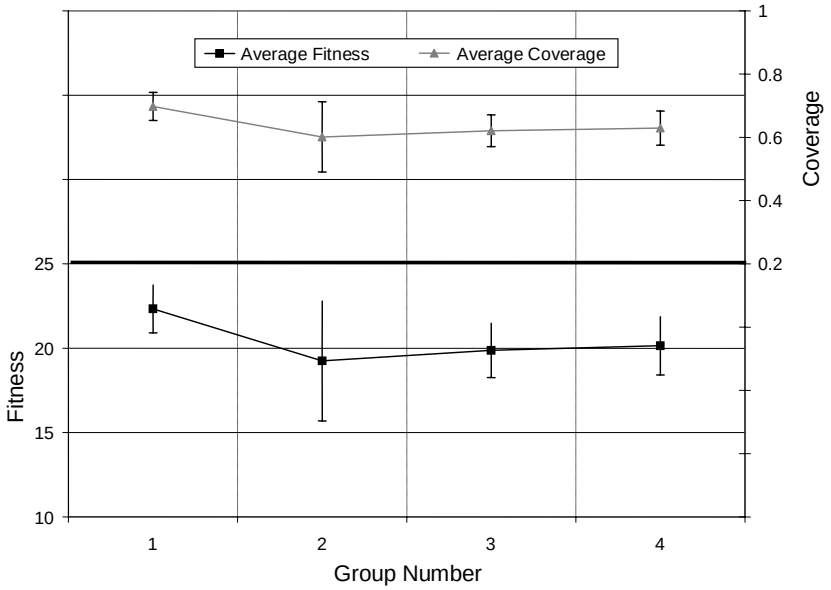


Figure 6. Coverage and fitness by group (results obtained from test configuration 2).

Each data point on this chart corresponds to the average of the best individuals created by the genetic algorithm at the end of each run. Notice that for both the average coverage and the average fitness, the group with the highest performance was Group 1.

How might one be able to improve the performance of the algorithm on this area? There are a few possibilities. One method of improving the results might be to increase the number of generations that the genetic algorithm is allowed to run. This could allow the genetic algorithm more time to create a better solution. Another method of improving the results might be to "seed" the initial population with a set of predefined paths, in effect giving the algorithm a "jump start" on a solution. These methods were not attempted for this project; however, they are interesting possibilities for further development.

## 4. CONCLUSIONS

This research was intended to conduct a feasibility study on whether it is possible to create a GA-based path planner for machinery operating on agricultural field. It has proven that it is feasible to use a genetic algorithm to create optimized paths through a field. Although this research did not create completely optimized paths, computer simulation analysis demonstrated over 90% coverage on fields of different shapes with various obstacles within the fields.

This research set the foundation for future development by setting up a framework for GA-based path planning system. This research also uncovered a few issues of worthy further investigation. For example, by making the genetic algorithm more intelligent and/or by adding constraints specific to the agricultural problem, it could possibly make the GA-based path planning algorithm capable of planning paths on uneven field terrains by represent the field in a three-dimensional format. An additional enhancement could be to design a "keep out" region around each obstacle, in which the path would not travel, thus ensuring that the vehicle would remain a safe distance from the obstacle. One constraint that was not utilized in this research was the use of vehicle parameters in planning paths. By including the vehicle parameters, such as the vehicle size, the turning radius and the maximum speed, it might help to generate paths more suited to agricultural vehicles to follow.

## 5. ACKNOWLEDGEMENTS

The material presented in this paper was based upon work supported partially by USDA Hatch Funds (ILLU-10-352 AE). Deere & Company supports Ms. Ryerson conducting graduate studies at the University of Illinois at Urbana-Champaign. Any opinions, findings, and conclusions expressed in this publication are those of the authors and do not necessarily reflect the views of the University of Illinois, Deere & Company and USDA.

## 6. REFERENCES

Capozzi, B.J., and J. Vagners. 2002. Evolution as a guide for autonomous vehicle path planning and coordination. In *Aerospace Conference Proceedings, IEEE* 5: 2527–2536. Piscataway, NJ, USA: IEEE.

Gerke, M. 1999. Genetic path planning for mobile robots. *In Proceedings of the 1999 American Control Conference* 4:2424-2429. Piscataway, NJ, USA: IEEE.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning* Reading, MA: Addison-Wesley.

Hocaoglu, C., and A.C. Sanderson. 1998. Multi-dimensional path planning using evolutionary computation. In *Proc. IEEE Conference on Evolutionary Computation,* 165-170. Piscataway, NJ, USA: IEEE.

Kambhampati, S., and L. Davis. 1986. Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation*, 2(3): 135-145.

Lin, H.-S., J. Xiao, and Z. Michalewicz. 1994. Evolutionary algorithm for path planning in mobile robot environment. In *Proc. IEEE Conference on Evolutionary Computation*, 1: 211-216. Piscataway, NJ, USA: IEEE.

Noguchi, N., and H. Terao. 1997. Path planning of an agricultural mobile robot by neural network and genetic algorithm. *Computers and Electronics in Agriculture* 18: 187-204.

Palmer, R.J. 1991. The adoption of automatic field operations. In *Proc. IEEE Western Canada Conference on Computer, Power and Communications Systems in a Rural Environment*, Piscataway, NJ, USA: IEEE.

Sugihara, K. 1997. A case study on tuning of genetic algorithms by using performance evaluation based on experimental design. *Technical Report ICS-TR-97-01, Dept. of Information and Computer Sciences*, University of Hawaii at Manoa.