Intelligent Information Management, 2018, 10, 1-15
http://www.scirp.org/journal/iim
ISSN Online: 2160-5920
ISSN Print: 2160-5912

Maze Navigation via Genetic Optimization

David J. Webb[1], Wissam M. Alobaidi[2*], Eric Sandgren[2*]

[1]Axalta Coating Systems, Front Royal, Virginia, USA
[2]Systems Engineering Department, Donaghey College of Engineering & Information Technology, University of Arkansas at Little Rock, Little Rock, Arkansas, USA
Email: *wmalobaidi@ualr.edu, *exsandgren@ualr.edu

How to cite this paper: Webb, D.J., Alobaidi, W.M. and Sandgren, E. (2018) Maze Navigation via Genetic Optimization. *Intelligent Information Management*, **10**, 1-15.
https://doi.org/10.4236/iim.2018.101001

Received: October 13, 2017
Accepted: December 12, 2017
Published: December 15, 2017

Abstract

One of the most interesting applications of genetic algorithms falls into the area of decision support. Decision support problems involve a series of decisions, each of which is influenced by all decisions made prior to that point. This class of problems occurs often in enterprise management, particularly in the area of scheduling or resource allocation. In order to demonstrate the formulation of this class of problems, a series of maze problems will be presented. The complexity of the mazes is intensified as each new maze is introduced. Two solving scenarios are introduced and comparison results are provided. The first scenario incorporated the traditional genetic algorithm procedure for the intended purpose of acquiring a solution based upon a purely evolutionary approach. The second scenario utilized the genetic algorithm in conjunction with embedded domain specific knowledge in the form of decision rules. The implementation of domain specific knowledge is intended to enhance solution convergence time and improve the overall quality of offspring produced which significantly increases the probability of acquiring a more accurate and consistent solution. Results are provided below for all mazes considered. These results include the traditional genetic algorithm final result and the genetic algorithm optimization approach with embedded rules result. Both results were incorporated for comparison purposes. Overall, the incorporation of domain specific knowledge outperformed the traditional genetic algorithm in both performance and computation time. Specifically, the traditional genetic algorithm failed to adequately find an acceptable solution for each example presented and prematurely converged on average within 54% of their specified generations. Additionally, the most complex maze generated an optimal path directional sequence (*i.e.* N, S, E, W) via a traditional genetic algorithm which possessed only 50% of the required allowable path sequences for maze completion. The incorporation of embedded rules enabled the genetic algorithm to locate the optimum path for all examples considered within 5% of the traditional genetic algorithm computation time.

Keywords

Maze Navigation, Genetic Optimization

1. Introduction

Decision support has developed into a broad spectrum of applications encompassing optimization through a variety of methods including genetic algorithms [1]. The traditional genetic algorithm is an evolutionary approach where problem characteristics are encoded to initially form random chromosome strings where strings are paired and the exchange of essential data is passed to create offspring. This offspring is evaluated against an objective function and potential optimization constraints which determine the success of the derived offspring. Additional key factors in the genetic algorithm evolutionary process include penalty factors which minimize the occurrence of poor offspring and mutation which randomly alters the encoded string to produce designs potentially unattainable within a small population size. Extensive background and theory regarding the fundamental methodology of the genetic algorithm can be found in [2] and [3].

The aim of this research is to illustrate the use of domain specific knowledge to enhance the genetic algorithm search through the minimization of computation time for solution convergence. Domain specific knowledge has been introduced into the genetic algorithm in the form of a two-phase rule approach to enhance both topological and structural optimization problems as demonstrated by Webb, *et al.* [4] and [5].

This study determines the appropriate path sequence to effectively negotiate a series of mazes within this prescribed research. Similarly to this optimization initiative, the use of domain specific knowledge was demonstrated by Alobaidi, *et al.* [6] to determine the optimal travel path sequences for the Traveling Salesman problem through the use of a rule based genetic optimization algorithm. Overall, the incorporation of a rule based enhancement to the genetic algorithm can be introduced within a variety of methods which includes a phased based rule encoding scheme as proposed by Sandgren *et al.* [7]. Phase one mimics the formulation of a traditional genetic algorithm string; however the second phase utilizes rule based chromosome strings to determine what domain specific knowledge or rule executes to ultimately improve the outcome of phase one. This research exposes embedded rules within the traditional genetic algorithm to further illustrate an alternative means for utilizing domain specific knowledge while enhancing the traditional genetic algorithm process.

2. Traditional Genetic Optimization Approach

Traditional genetic optimization or scenario one began with a series of encoded

strings which represents the framework of a maze under investigation. These encoded strings allow the genetic algorithm to determine whenever a move is selected (*i.e.* N, E, S, W) that the move is a legitimate move within the maze. A maze essentially is a predefined amount of space where a grid or lattice is formed which represents a series of rows and columns of square blocks. Each block formed within the lattice represents a "cell". The surrounding walls of user specified cells are removed and the formation of a maze becomes apparent. The strings within this input file are four characters long which represent the directions of north, east, south and west respectively. Each character within the string can either represent the values of "0" or "1" to appropriately define each cell. A value of "0" for the first character within the first row indicates that there is a north wall within the first cell of the grid. Conversely, a value of "1" for the first character within the first row indicates an absence of a north wall within the first cell of the lattice. Each cell is labeled throughout the maze where the first block or entrance of the maze is designated by (1, 1), meaning row one, column one, and is located at the lower left corner as illustrated by the example in Figure 1. The exit or last cell for the maze illustrated in Figure 1 is numerically identified by (10, 10) which is the upper right corner of the maze. Strings within the initial maze input file are read in left to right and row by row beginning with the first cell block. All mazes investigated began with the starting cell of (1, 1) and exited at the last cell of the grid. These constant start and end locations were imposed since this generated the longest and most difficult search solution possible for the genetic algorithm. The objective was to locate the travel route which connected the start cell with the end cell. The function of the genetic algorithm was to generate an initial travel sequence string based upon travel directions of north, east, south, and west. The amount of characters which composed of a maximum travel sequence was equal to the number of cells generated by the implementation of a grid. Each character within the travel sequence is assigned a numeric value within a range of four possible values. Table 1 provides the possible values for which a character within the travel sequence string could potentially represent. Additionally, Table 2 illustrates the genetic algorithm parameters

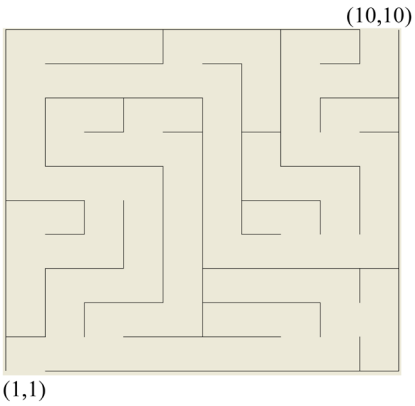

Figure 1. 10 × 10 grid maze.

Table 1. Character representation values for travel sequence string.

Character Value	Representation
1	North
2	East
3	South
4	West

Table 2. Genetic input parameters.

GENETIC INPUT PARAMETERS
Population Size = Maze Grid Size * 10
Probability of Mutation = 0.02
Number of Generations = (5 * Maze Grid Size) + Number of Best Generation Members to Pass to the Next Generation
Penalty Factor = 10
Multiplier of Penalty Factor = 2
Number of Best Generation Members to Pass to the Next Generation = 2
Number of Generations Between Penalty Factor Updates = 10

utilized in the problem formulation for all mazes considered.

A random population of travel sequences is generated and evaluated by the objective function as illustrated in Equation (1). If deemed a legitimate move within the maze, the objective function calculates the total distance of each travel sequence, however the final result is path and distance weighted as illustrated in Equation (2), for the directional moves within the travel sequence are theoretically legitimate, but actual distances may not be calculated in the event N-S or E-W moves are introduced into the travel sequence. Once evaluated by the objective function, the best travel sequences are selected and placed within a mating pool, where travel sequences are randomly paired and genetic chromosome cross over commences. Chromosome cross over is the process of mating two parent travel sequence strings by the random selection of a numbered character shared by both strings. Once a shared character between both strings is selected, each chromosome string exchanges characteristics the right of the selected gene until the end of both chromosome strings has been reached. Subsequent to genetic cross over, each offspring produced is evaluated by the objective function, where directional moves within the new travel sequence are only implemented if legitimate within the maze. Illegitimate moves within travel sequences are simply ignored by the objective function and the travel sequence string is reduced in total length. This process continues until a predetermined number of generations of offspring have been produced. Mutation, or the random alteration of genes of a parent travel sequence string within the mating pool occurs, however the probability is minimal. Mutation introduces randomness, which provides the

design or decision chromosomes with characteristics normally unattainable subsequent to a number of generated offspring. Upon the completion of a user specified number of generations, the genetic algorithm has potentially created a travel sequence string which encompasses the total or partial path which unites both start and end cells.

Objective Function:

$$f(x)=\sqrt{\left(\left(x_{\text{actual}}-x_{\text{final}}\right)^2+\left(y_{\text{actual}}-y_{\text{final}}\right)^2\right)} \tag{1}$$

Weighted objective function $f(x)_{\text{Weighted}}$

$$f(x)_{\text{Weighted}}=W1-W2*\left(\sqrt{\left(\left(x_{\text{actual}}-x_{\text{final}}\right)+\left(y_{\text{actual}}-y_{\text{final}}\right)^2\right)}-W3*\left(npath-C\right)\right) \tag{2}$$

where, $npath=$ total number of paths each cell move possesses that culminates the defined path.

x_{actual}, y_{actual}, x_{final}, & y_{final} are parameters which represent the start and end point locations of travel sequences for each maze considered which determine overall travel distance

$W1$, $W2$, & $W3$ are relevant weighting factors to balance out the outcome of the objective function with the number of moves required utilizing constant factor C. The use of a weighted objective function accounts for directional moves within the travel sequence; however actual distance values may not be calculated in the event N-S or E-W moves are introduced into the travel sequence

3. Genetic Optimization via Embedded Rules

Domain specific knowledge in the form of rules were embedded within scenario one. These rules are governed by each travel sequence and the travel sequence is controlled by the genetic algorithm.

The cells which compose of the maze are initially predefined by how many exits each cell possesses and if the cell possesses only one exit, information is provided to which direction the available exit is located (*i.e.* N, E, S or W). When a cell is located within the travel sequence and only one exit is available, the genetic code is provided with what direction out the cell possesses. With this information, a wall is created to block off the cell, creating a block, so that no future travel sequences may enter. All cells are updated with the creation of this new wall and the process continues whenever a travel sequence encounters a one exit cell. This prescribed embedded rule is attributed by prior knowledge acquired from Williams [8]. Additionally, when the objective function evaluates each move within a specific travel sequence, redundant moves are removed such as N-S and E-W, so that every move accepted creates an actual distance.

The implementation of rules within the objective function subroutine which eliminate redundant moves before being drawn is controlled by the genetic code, since the travel sequence is governed by the genetic algorithm and rules are applied as travel sequences are introduced to the embedded rules. This also holds

true for walls created when travel sequences encounter a cell with only one exit. The rule for the implementation of a wall is only executed if the genetic code generates a travel sequence which leads to a cell with only one exit. Computation time for solving all mazes considered was minimal with the inclusion of embedded rules.

4. Results

Three mazes were investigated and solved by the use of both a traditional genetic algorithm and a genetic algorithm with embedded rules. Each maze scenario presented was programmed in Visual Basic [9]. Genetic algorithm parameters utilized in the formulation of the provided results are illustrated in **Table 3**. The mazes constructed were 10 × 10, 20 × 20, and 50 × 50 cell mazes, which demonstrated a gradual progression in maze complexity as each maze was examined and solved. Overall, the genetic algorithm which incorporated embedded rules was the most effective optimization approach. The injection of domain specific knowledge into a genetic algorithm significantly enhanced computation time and the quality of genetic offspring produced. Genetic results for both optimization approaches initially appear at the end of the second generation, for both optimization techniques generated new offspring travel sequence strings immediately following the construction of the initial random population. The generation of new offspring subsequent to the creation of an initial random population was implemented to immediately reduce any extraneous travel sequence strings from the random population to potentially improve genetic offspring and computation time. Lastly, graphical representations of objective function versus generation were constructed for each maze solving procedure considered.

Function evaluations considered are derived via Equation (3) below

$$\text{Function Evaluations} = \text{Population}_{\text{Size}} * \text{Generation}_{\text{Size}} \tag{3}$$

5. Example One: 10 × 10 Cell Maze

Below illustrated in **Figure 2** is the initial 10 × 10 cell maze before the commencement of genetic optimization. Notice that the complexity of the maze is fairly straightforward, but results below confirm that a conventional genetic algorithm was unable to locate an acceptable solution.

5.1. Traditional Genetic Algorithm Result (Scenario One)

The traditional genetic algorithm result is shown below in **Figure 3**. Upon the

Table 3. Maze genetic parameters.

Maze Grid Size	Population Size	Generation Size	Function Evaluations
10 × 10	100	52	5200
20 × 20	200	102	20,400
50 × 50	500	252	126,000

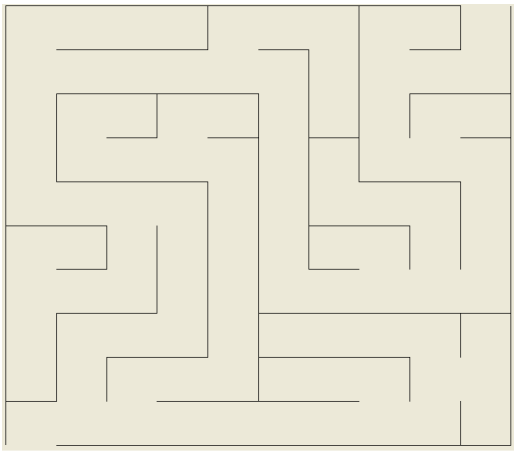

Figure 2. 10 × 10 cell maze.

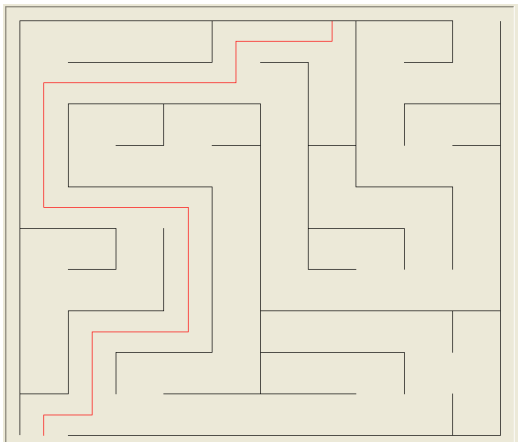

Figure 3. Genetic algorithm result.

completion of fifty two generations and an initial population size of one hundred, the genetic algorithm was unable to locate an acceptable solution. Population and generation sizes must be largely increased to potentially locate the maze solution. The objective function versus generation graph provided in **Figure 4** revealed an overall steady increase in objective function until generation forty five and remained constant throughout the remainder of the fifty two specified generations. An increase in these genetic input parameters would significantly hinder computation time, and would further reinforce the necessity to refine the genetic algorithm procedure. Theoretically, once a considerable number of generations of offspring have been produced an acceptable solution would be achieved. However, genetic input parameters were increased on several occasions, but an acceptable solution was still not acquired.

5.2. Genetic Algorithm with Embedded Rules Result (Scenario Two)

Below illustrated in **Figure 5** is the final result of the 10 × 10 cell maze which was solved by the use of a genetic algorithm with embedded domain specific

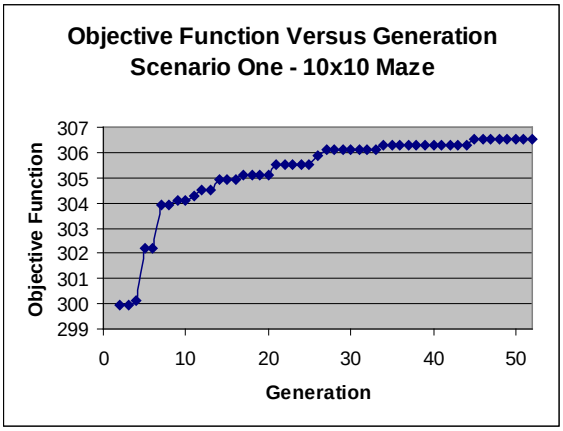

Figure 4. Objective function versus generation.

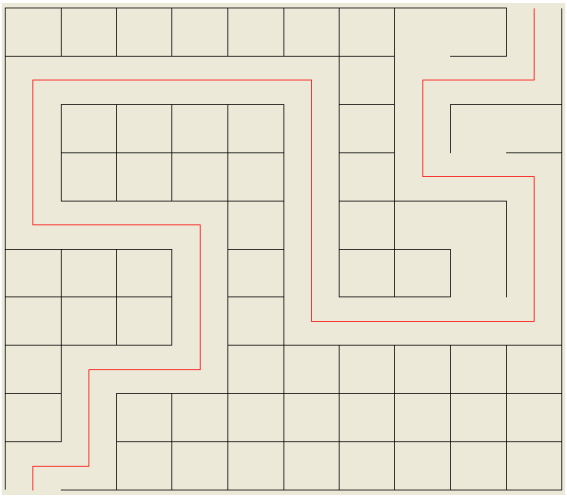

Figure 5. Embedded rules final result.

knowledge. Genetic input parameters were identical to the traditional genetic optimization result illustrated in **Figure 3**. An examination of **Figure 5** revealed that not all travel paths were blocked, which clearly indicated for even the least complex of mazes that the embedded rules are travel sequence dependent, where each travel sequence is controlled by the genetic algorithm. Furthermore, the objective function versus generation graph illustrated in **Figure 6** revealed that a solution was found upon the conclusion of the second generation of offspring produced. In comparison to **Figure 4**, the traditional genetic optimization approach spawned fifty two generations of offspring, yet was incapable of generating an acceptable solution. Clearly this approach is the optimal method for solving this class of problems.

6. Example Two: 20 × 20 Cell Maze

Figure 7 represents the 20 × 20 cell maze before genetic optimization has commenced. In comparison to the previously illustrated 10 × 10 maze, the complexity has significantly increased, which requires larger genetic input parameters for

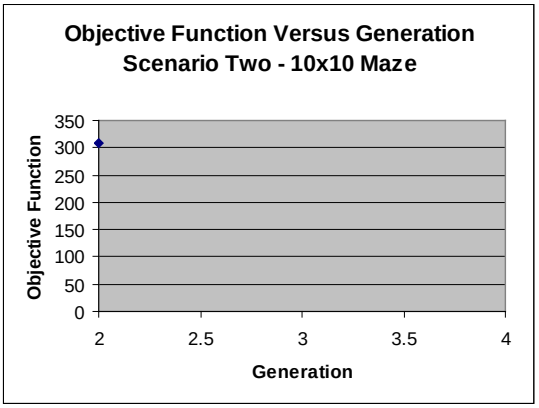

Figure 6. Objective function versus generation.

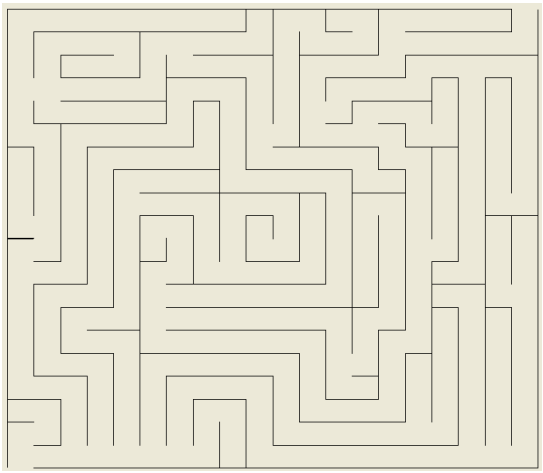

Figure 7. 20 × 20 cell maze.

the traditional genetic algorithm technique, for the genetic algorithm is a global search method based upon an evolutionary approach.

6.1. Traditional Genetic Algorithm Result (Scenario One)

Illustrated within **Figure 8** is the traditional genetic algorithm result. Genetic input parameters consisted of a population size of two hundred with one hundred-two generations. Notice that an acceptable solution was unable to be located upon the conclusion of one hundred-two generations. Population and generation genetic input parameters were manipulated on several instances but, an acceptable solution remained unachievable.

An examination of **Figure 9** revealed a steady increase in objective function over a period of eighty generations, however inactivity was apparent throughout the remaining twenty two generations.

6.2. Genetic Algorithm with Embedded Rules Result (Scenario Two)

The illustration within **Figure 10** represents the final result which utilized the

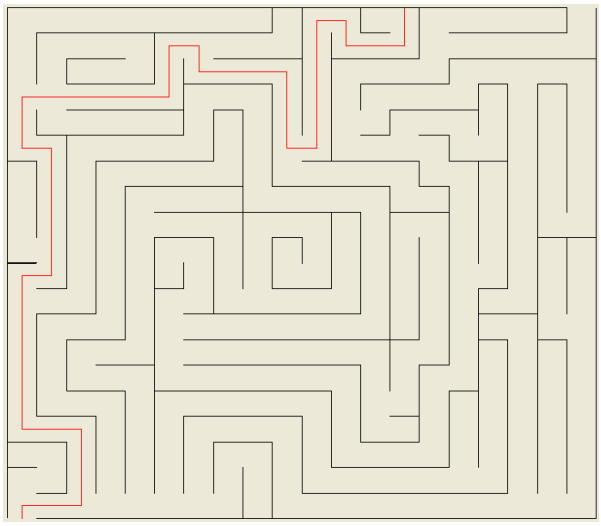

Figure 8. Genetic algorithm result.

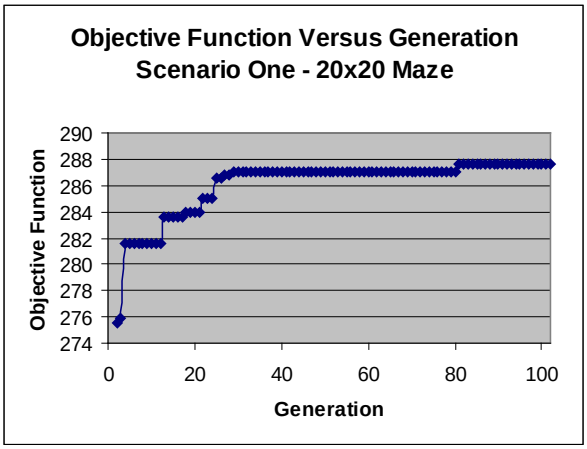

Figure 9. Objective function versus generation.

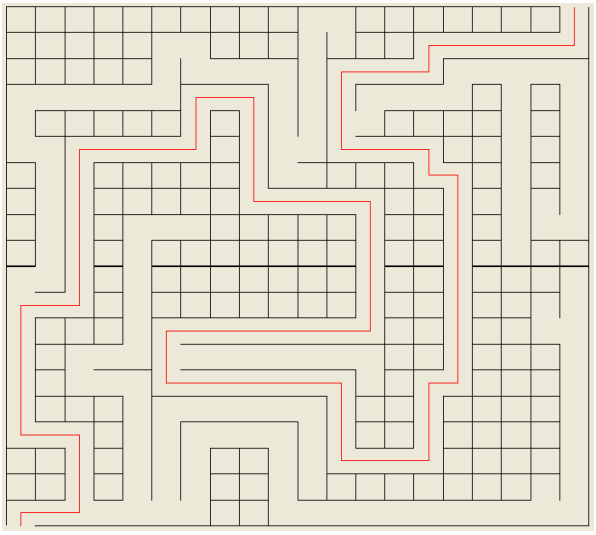

Figure 10. Embedded rules final result.

genetic algorithm with an embedded rules approach applied to the 20 × 20 cell maze. Notice that all routes were not blocked, for the genetic algorithm found the optimal path without traveling to every available location, which further reinforced that the rules embedded are executed based upon travel sequences controlled by the genetic algorithm. Figure 11 revealed that a solution was found within the second generation of offspring produced. A comparison with the result illustrated in Figure 8 showed that the traditional genetic algorithm solution generated a final result which was incomplete, and furthermore failed to locate the correct travel path concluding one hundred-two generations of offspring. Failure to locate the correct travel path upon the conclusion of the conventional genetic algorithm method further reinforces the necessity of a rule based scenario to enhance the traditional genetic algorithm. Potentially large phase two population and generation sizes would be considered to further refine this conventional genetic algorithm result, for a vast variety of rule sequences must be generated to correct the inadequately generated travel path.

7. Example Three: 50 × 50 Cell Maze

The final maze investigated consisted of a 50 × 50 cell maze [10]. This maze incorporated the highest level of complexity among the mazes previously examined. Twenty five hundred cell blocks were manipulated in the formation of the maze illustrated below in Figure 12. The ability to utilize domain specific knowledge within a genetic algorithm to enhance solution time and the quality of offspring generated, while demonstrating that an acceptable solution can be located at this level of complexity, settles any disputes with regard to the need for rules within a genetic algorithm.

7.1. Traditional Genetic Algorithm Result (Scenario One)

The traditional genetic algorithm approach was unable to locate an acceptable solution shown in Figure 13 however unlike the 20 × 20 cell maze traditional

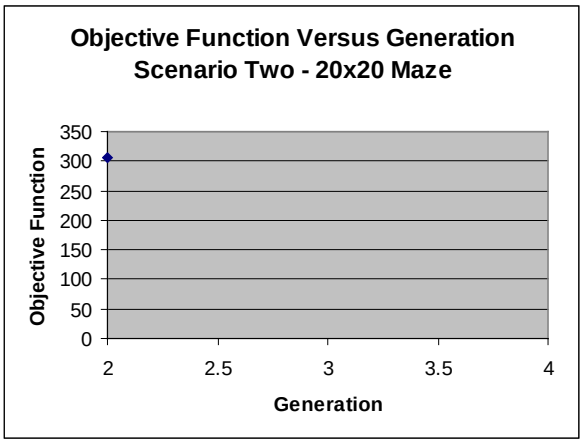

Figure 11. Objective function versus generation.

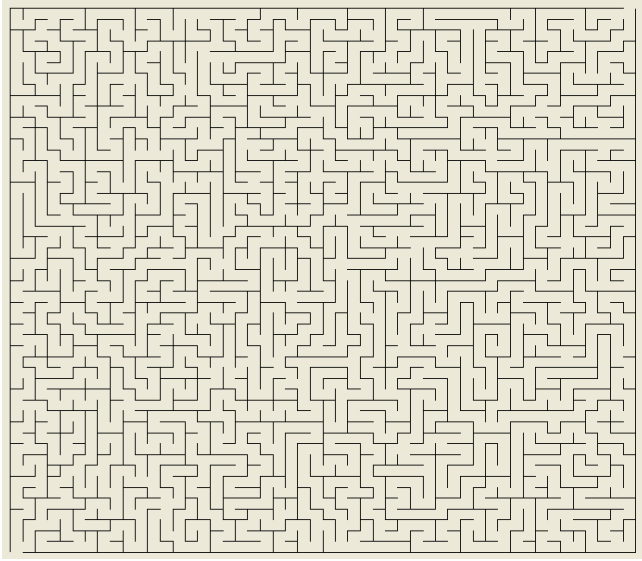

Figure 12. 50 × 50 cell maze.

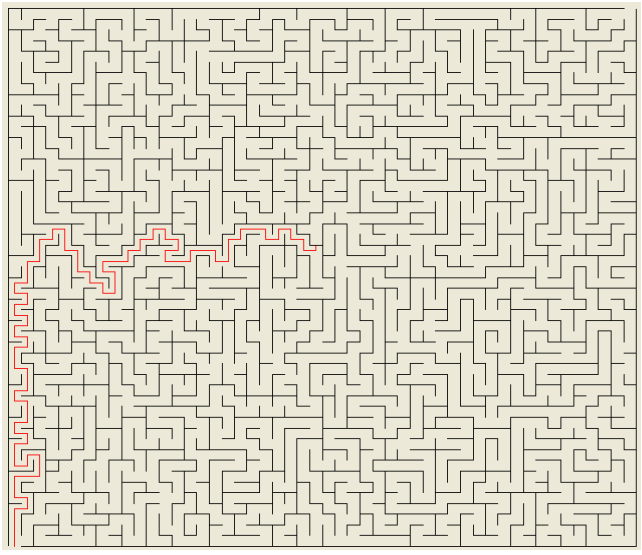

Figure 13. Genetic algorithm result.

genetic algorithm result; the genetic algorithm was able to locate a portion of the optimal path. Genetic input parameters consisted of a population size of five hundred and a generation size of two hundred fifty-two. Since an acceptable portion of the path was located, an increase in genetic input parameters would potentially improve the probability of locating a more acceptable solution in comparison to the previous 20 × 20 and 10 × 10 maze results. Surprisingly, the conventional genetic algorithm procedure failed to locate the complete travel path for each maze considered.

An examination of **Figure 14** shows a steady but minimal increase in objective function over a period of one hundred eighty-seven generations, yet remained inactive throughout the remaining sixty five generations.

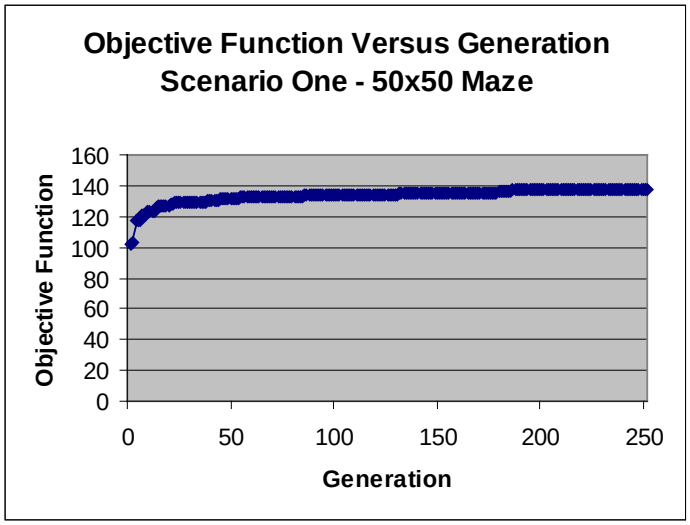

Figure 14. Objective function versus generation.

7.2. Genetic Algorithm with Embedded Rules Result (Scenario Two)

The genetic algorithm which incorporated embedded rules was utilized to solve the 50 × 50 cell maze. Genetic input parameters remained identical to its counterpart. This optimization approach effectively solved the maze illustrated in **Figure 15**. Similar to previous results, this optimization approach successfully located the maze exit without traveling to every possible location within the maze, but only by travel sequences created by the genetic algorithm. The objective function versus generation graph within **Figure 16** revealed that the final solution was located upon the completion of four generations of offspring. A comparison with **Figure 14** clearly indicates that the incorporation of embedded rules significantly decreases computation time and enhances the quality of genetic offspring produced, since only four generations of offspring were required to achieve a solution, where two hundred fifty-two generations were completed using a traditional genetic algorithm approach, yet a complete solution remained unattainable.

8. Concluding Remarks

The implementation of embedded rules within a genetic algorithm based upon domain specific knowledge significantly decreases computation time and enhances the quality of offspring produced during each generation. A complete and accurate solution was located for all mazes investigated which employed the use of domain specific knowledge within a conventional genetic algorithm. The traditional genetic algorithm approach failed to solve each maze provided, while subjected up to extensively larger generations than its counterpart approach. Lastly, the genetic algorithm with embedded rules required only 5% of the solution time for all mazes considered in comparison to the traditional genetic algorithm

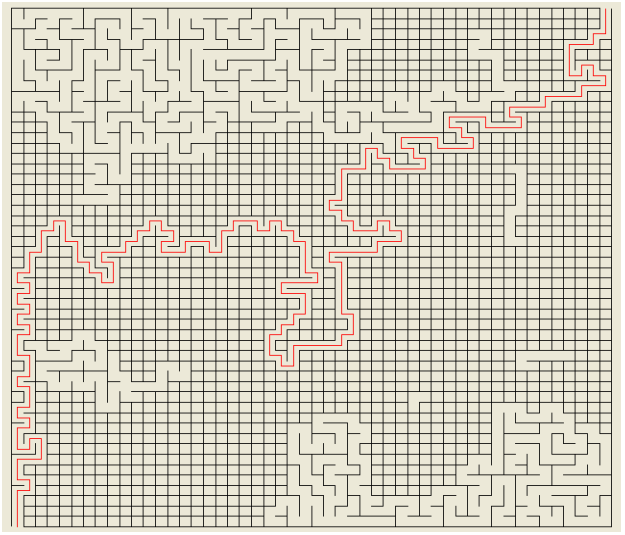

Figure 15. Embedded rules result.

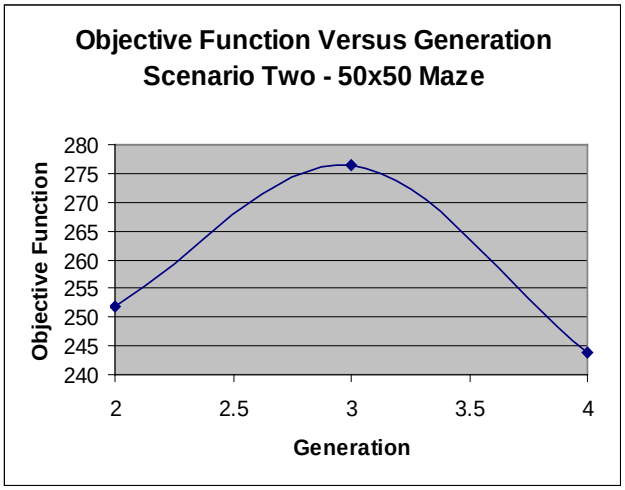

Figure 16. Objective function versus generation.

to locate an acceptable solution which effectively solved each maze.

References

[1] Alobaidi, W., Sandgren, E. and Alkuam, E. (2017) Decision Support through Intelligent Agent Based Simulation and Multiple Goal Based Evolutionary Optimization. *Intelligent Information Management*, **9**, 97-113. https://doi.org/10.4236/iim.2017.93005

[2] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

[3] Gen, M. and Cheng, R. (1997) Genetic Algorithms & Engineering Design. John Wiley & Sons, Inc., New York.

[4] Webb, D., Liu, Q., Alobaidi, W. and Sandgren, E. (2017) Topological Design via a Rule Based Genetic Optimization Algorithm. *American Journal of Computational Mathematics*, **7**, 291-320. https://doi.org/10.4236/ajcm.2017.73023

[5] Webb, D., Alobaidi, W. and Sandgren, E. (2017) Structural Design via Genetic Optimization. *Modern Mechanical Engineering*, **7**, 73-90. https://doi.org/10.4236/mme.2017.73006

[6] Alobaidi, W., Webb, D. and Sandgren, E. (2017) A Rule Based Evolutionary Optimization Approach for the Traveling Salesman Problem. *Intelligent Information Management*, **9**, 115-132. https://doi.org/10.4236/iim.2017.94006

[7] Sandgren, E., Webb, D. and Pedersen, J.B. (n.d.) A Rule Based Evolutionary Algorithm for Intelligent Decision Support.

[8] Williams, S. (2004) Maze Solving. http://www.undu.com/Articles/000229a.html

[9] Microsoft Corporation (1998) Microsoft Visual Basic Version 6.0.

[10] Mazes (2004). http://www.mazes.org.uk/