Intelligent Information Management, 2017, 9, 115-132
http://www.scirp.org/journal/iim
ISSN Online: 2160-5920
ISSN Print: 2160-5912

A Rule Based Evolutionary Optimization Approach for the Traveling Salesman Problem

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

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

How to cite this paper: Alobaidi, W.M., Webb, D.J. 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

Received: June 7, 2017
Accepted: July 11, 2017
Published: July 14, 2017

Abstract

The traveling salesman problem has long been regarded as a challenging application for existing optimization methods as well as a benchmark application for the development of new optimization methods. As with many existing algorithms, a traditional genetic algorithm will have limited success with this problem class, particularly as the problem size increases. A rule based genetic algorithm is proposed and demonstrated on sets of traveling salesman problems of increasing size. The solution character as well as the solution efficiency is compared against a simulated annealing technique as well as a standard genetic algorithm. The rule based genetic algorithm is shown to provide superior performance for all problem sizes considered. Furthermore, a post optimal analysis provides insight into which rules were successfully applied during the solution process which allows for rule modification to further enhance performance.

Keywords

Traveling Salesman, Evolutionary Optimization, Rule Based Search, Heuristic Optimization, Hybrid Genetic Algorithm

1. Introduction

The traveling salesman problem is an example of a NP complete problem where the computational time required to generate an exact solution increases exponentially with the number of cities involved. The objective is to minimize the path length required to visit a specified set of N cities, starting at one city, visiting each city exactly one time and returning to the city from which the path was originated. This problem is known as a combinatorial minimization problem

with discrete variables. The discrete nature of the problem arises from the fact that each city may be numbered as an integer selection and a non-integer selection has no significance. The number of possible routes is factorially large, so that the solution may not be generated practically via an exhaustive search. The discrete nature of the problem eliminates the use of gradient nonlinear programming techniques as well as introducing a large number of local minima. The selection of cities without replacement (each city is visited only once) adds another difficulty in generating the solution through the use of a traditional genetic based algorithm. The problem's origin dates back to the early days of linear programming and continues to serve as a benchmark for new solution algorithms. The problem is representative of a large number of practical optimization formulations, including electronic circuit design, scheduling, pick-up and delivery and providing home health care or other services. The size of practical applications can range from tens of cities to tens of thousands of cities. While many algorithms can solve problems involving tens of cities, most have extreme difficulty with problems involving over one hundred cities.

A host of solution algorithms have been developed to address the traveling salesman problem over a time period of more than fifty years starting with the pioneering work of Dantzig [1]. The original work treated the problem as a discrete linear programming problem and was severely limited in the number of cities that could be addressed. Since that time, applications of various branch and bound methods [2] have been applied to handle the discrete nature of the problem. Simulated annealing [3] [4] approaches have been utilized in order to avoid the multitude of local minima contained in the problem. Tabu search [5] and other meta-heuristic approaches have been applied as another means of locating a global solution. Neural networks applications provide one means of implementing a simplistic rule based solution [6]. Ant colony search [7] provides another approach which seeks to mimic natural processes to avoid being trapped at local minima. Various versions of evolutionary optimization [8] [9] have been developed for the particular problem class represented by the traveling salesman problem. An excellent summary of approaches prior to 1983 is provided by Kindervater and Lenstra [10]. Summaries of more modern approaches are common in the literature [11] [12] [13]. The search for a reliable solution technique continues today with recent work by Agarhor *et al*. [14] utilizing an improved genetic algorithm based on the behavior of predatory animals, and the previous work by Kaur and Murugappan [15] who produced a hybrid genetic algorithm which works on combinations of nearest neighboring cities in conjunction with a traditional genetic algorithm. Much progress has been made over the years, but to date generating an optimal solution to the traveling salesman problem remains a difficult task. It remains the topic of considerable interest.

An interesting collective attribute found in most of the approaches is the application of heuristic procedures imbedded in various aspects of the solution process. These procedures are found to work well for certain classes or subsets of

the traveling salesman problem, but few algorithmic platforms exist which can implement a wide range of heuristics in an intelligent environment. A rule based, evolutionary approach has the compatibility of supporting a general heuristic environment where the success of each heuristic can be monitored and the overall performance of the algorithm can be improved over time through continued monitoring and refinement of the heuristics implemented.

The goal here is to demonstrate that a rule based genetic algorithm operating with a simplistic rule set can perform as well or better than an expert, which in this case will be represented by an algorithm explicitly designed for this class of problems, the method of simulated annealing. A brief review of the simulated annealing approach and a conventional genetic algorithm will lead into the development of the rule based approach. The three algorithms are then executed on sets of randomly generated traveling salesman problems ranging in size from ten to one hundred cities. Each problem size is represented by ten problems where the location of each city is generated randomly within the defined solution space. All algorithms are tasked with solving the same set of problems. None of the algorithms tested are claimed to be overly efficient, but the results represent the general trend which would be expected in the application of a generic implementation of the particular algorithm class. The results from this comparative study show the initial promise of the rule based approach. To further document the power of the approach, a set of fifteen problems taken from TSBLIB 95 [16] were also tested. This problem set includes problems collected specifically designed to test new solution methods for the traveling salesman problem.

From the work conducted to date, many of the most promising approaches to solving the traveling salesman problem have involved the use of heuristics. The ability to imbed an arbitrary set of heuristics into the framework of a genetic algorithm forms the basis of this research. Rules generated from previous hybrid methods have been combined with new strategies to form an efficient solution algorithm. This algorithm is tested on a wide variety of problems to demonstrate the ability of the approach in generating the global optimum for problems. While it is fairly easy to generate a local solution that is within five to ten percent of the global optimum, it is extremely difficult to generate the global optimum. In all test cases utilized, the global optimum was located. It is also demonstrated that by tracking the success of the various heuristics or rules, the algorithm can undergo continuous modification to increase the performance. This can lead to an automated rule update which would allow for an aspect of learning occurring.

2. Simulated Annealing

Simulated annealing is a global optimization technique which mimics the behavior of cooling metal from a molten to a solid state. At high temperatures, the molecules are free to move freely in the molten metal. As the liquid cools and solidifies, however, the mobility of the molecules is lost. If the process is carefully

controlled and the cooling takes place in a relatively slow fashion, a pure crystalline structure is formed. This structure represents the state of minimum energy. If the cooling is allowed to occur too rapidly, the material ends up in an amorphous state having higher energy in the structure. Most optimization algorithms are greedy in that they attempt to reach the minimum in the least amount of time. This corresponds to the quick quenching of a molten metal and generally results in the location of local minima. Following the analogy of the simulated annealing technique, a non-greedy or global optimization technique can be generated which is ideally suited for the traveling salesman problem.

The simulated annealing method is based on the Boltzmann probability distribution which expresses the probabilistic distribution of energy states for a system in equilibrium. The distribution may be expresses as:

$$\text{Prob}(E) \sim \exp\left(-\frac{E}{KT}\right) \tag{1}$$

In the above equation, T represents the system temperature, E, the system energy and k, the Boltzmann constant which is a physical constant. The equation states that even at a low temperature, there is a probability that the system may be in a high energy state. It is therefore possible to leave a local minimum energy state and find a lower energy state, although it may be necessary to temporarily increase the energy state to accomplish this. In the early 1950s, this principle was incorporated into a numerical optimization algorithm, known as the Metropolis algorithm [17]. From a current objective function value of E_1, a second value, E_2 will be accepted with a probability of

$$P = \exp\left(\frac{-(E_2 - E_1)}{KT}\right) \tag{2}$$

If E_2 is less than E_1, the probability is greater than unity and the new point is accepted. Even if E_2 is greater than E_1, there is a finite possibility of acceptance. This provides a general downhill search, with an occasional uphill move to help increase the likelihood of locating the global minimum. The values of K and T help define a specific algorithm. As the temperature T is reduced, the possibility of accepting a design which is inferior to the current design decreases.

To actually implement a Metropolis algorithm, several elements must be defined. These elements include:

1. A representation of possible system configurations
2. A random generator of new system configurations
3. An objective function to be minimized which can be calculated directly from a system configuration
4. A control parameter, T, and an annealing schedule which specifies how the temperature is lowered during the search process.

Some problem specific experimentation is generally required in order to determine appropriate values of T and the cooling schedule consisting of the number of iterations taken between temperature changes and the amount of a

temperature change. An implementation for the traveling salesman problem is straightforward.

Given a series of city location coordinates (x_i, y_i) for a total of N cities, the task becomes one of determining the order of travel from one city to the next while visiting each city exactly once and returning to the originating city by traveling the minimum total distance. The representation of possible system configurations is given by selecting a set of N integers without replacement which represents a possible route for the salesman. New configurations may be generated in a large number of ways. The particular generator used in this study is given in Numerical Recipes [18]. Two types of city rearrangements are considered. The first selects a portion of the route, removes it and replaces the path with the same cities in reverse order. The second rearrangement removes a portion of the path and inserts the removed path in a different location. These rearrangements were suggested by Lin [19]. They are of interest in this study as they are actually rule based modifications which can easily be implemented within the rule based genetic code. The objective function is simply the total distance traveled which in this case will be:

$$F(x) = \sum_{i=1}^{N} \left\{ \left(x_i - x_{i+1}\right)^2 + \left(y_i - y_{i+1}\right)^2 \right\}^{\frac{1}{2}} \tag{3}$$

It is understood that the $N + 1$ point is the origination city (city 1). The annealing schedule used is that suggested in the text Numerical Recipes. A starting temperature, T, is selected which is larger than any change in distance normally encountered during a reconfiguration. Each temperature is held constant for 100 N reconfigurations or after 10 N successful reconfigurations, whichever occurs first. The temperature is then decreased by ten percent and the process repeated until no improvement is made during the current iteration.

3. Solution via a Traditional Genetic Algorithm

In order to gauge the difficulty of solution of the traveling salesman problem, a traditional genetic algorithm was also utilized. Some modification was required in the design encoding and crossover operations to guard against the introduction of duplicate cities appearing in a specific design representation. As with the simulated annealing technique, a design representation consists of N unique values, with each value representing a city to visit. The order of the string of values signifies the order of travel which allows the total distance traveled to be calculated via Equation (3). For example a design representation for a ten city problem could be:

X = {1, 4, 7, 3, 5, 9, 2, 6, 10, 8}

Or

X = {7, 4, 2, 10, 3, 9, 1, 5, 8, 6}

Each representation consists of ten unique cities to visit in the order specified. The difficulty for the genetic algorithm is that there is no inherent way of re-

stricting the re-use of cities (*i.e.* selection without replacement). The second issue involves the crossover operation. As an example, let the two design representations listed above be the selected parents for a crossover operation. In addition, let the crossover point be given as the fifth position in the string. Switching the first and second portions of the design representations at the fifth position produces the following offspring:

$X_1 = \{1, 4, 7, 3, 5, 9, 1, 5, 8, 6\}$

And

$X_2 = \{7, 4, 2, 10, 3, 9, 2, 6, 10, 8\}$

Notice that even though both parent representations had no duplication of cities, both child representations do have multiple replications. Thus, the child representations do not represent valid travel orders for the problem. There are a number of possible implementations to avoid this problem. The one selected for this trial was simply to let each original design encoding to be represented by a string of integers, each ranging in value from one to N. Duplicate city values are eliminated during the distance evaluation by simple replacement of the duplicate values with the nearest (in number) city which has not been used previously in the design encoding. For example, the order represented by X_1 above would be evaluated as the string:

$X_1' = \{1, 4, 7, 3, 5, 9, 2, 6, 8, 10\}$

This replacement scheme also eliminates the crossover issue as duplicate values in the design representation are eliminated before evaluating the objective function. With this modification, the remainder of the genetic algorithm remained as coded for general problem solution.

4. The Rule Based Genetic Code

As opposed to a traditional genetic formulation, a rule based formulation utilizes an encoding which contains rules which operate on one or more trial orderings of cities. A single rule, or a combination of multiple rules, may be executed at any point in the solution process. As the process continues, the trial orderings improve and a history of which rules or combination of rules were successful in the search is maintained. This procedure converts the genetic algorithm to a heuristic approach which is more in line with algorithms which have proven to be capable of solving the traveling salesman problem. The maintained history may be utilized to improve the rule set by eliminating rules which had little impact on the process and continually improving the rule which were utilized successfully. A rich set of potential rules is available from the wide variety of heuristic algorithms generated to date. A major advantage of the rule based approach is that the encoding size utilized in the algorithm need not increase with the size of the problem being solved.

At an elementary level, the rule based evolutionary process may be defined by an encoding of a rule set similar to that shown in **Figure 1**. Here, the first element in the encoding string identifies how many rules are to be executed. This

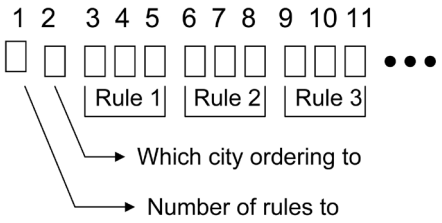

Figure 1. A rule based encoding for the traveling salesman problem.

allows for several rules to be applied to a trial ordering at the same time. The second element in the decision string specifies which current trial ordering of cities to apply the rules to. If only a single trial ordering is maintained, this element may be eliminated. Realizing the fact that there are a multitude of local minima in the search space, it seems wise to operate on a population of trial city orderings. This may increase the solution time, but allows for a population of rules to be applied to a population of trial orderings. Subsequent groups of encoding elements are utilized to define which specific rule or rules to apply with specific information blocks which define precisely how each rule is to be executed. The fact that only three such rule execution blocks are included in the encoding represented in **Figure 1** is not a limitation as the encoding may be expanded as needed or desired. In order to insure consistency in the crossover operation, the length of each rule execution block equal in length.

For the rule based genetic code, five separate rules were implemented. These rules are described as follows:

1. Selecting a group of cities in the design representation, removing them and inserting the group in an alternate location.
2. Selecting the closest neighbors to a specific city and exchanging the position of the nearest neighbors with the current neighbors.
3. Ordering a selected subgroup of cities by relative distance to each other.
4. Selecting a group of cities and reversing the order of the cities in that group.
5. Reordering a sub-group of cities based on roulette wheel selection based on distance.

Rules one and four are simply the rules implemented in the simulated annealing code. Rules two and three are distance re-ordering procedures as is rule five which actually makes use of coding present in the genetic algorithm. Rule five is specifically inserted to allow for a route selection which is not simply the local least distance from city to city to help avoid the arrival at a local minimum. Other rules could have been selected and perhaps would have been more appropriate. The goal here is to simply demonstrate the effectiveness of the concept rather than to develop the ultimate genetic algorithm for the traveling salesman problem.

The encoding of the rules in a design encoding is contained in the generic format listed below:

$$ABC_1D_1E_1F_1\ C_2D_2E_2F_2\ C_3D_3E_3F_3\cdots C_nD_nE_nF_n \tag{4}$$

The A field represents the number of rules to apply, the B field represents which of the current city orderings to apply the rule set to. The field C_i represents which rule to apply and the fields D_i, E_i and F_i provide specific rule implementation information. The subscript n represents the maximum number of rules to be exercised in one modification of a selected design encoding (7 for this trial). The values for the B string position are integers in the range of one to the number of trial city orderings retained in the search process. The range of values for string position C_i includes the integers from one to the number of possible rules (5 in this example). The values of the remaining string positions are interpreted according to the rule specified by the C_i string position. For this particular implementation, the fields D_i, E_i and F_i represent city location or design string location and as such can have any integer value from 1 to N, the number of cities to visit. The specific interpretation of these values is defined as:

For rule 1:

D_i represents the first city in the selected group for movement in the design string.

E_i represents the number of cities in the selected group for movement.

F_i represents the city after which the selected string is inserted.

For rule 2:

D_i represents the first city in the selected group to re-order by distance.

E_i represents the number of cities to re-order by distance.

F_i is ignored for this rule implementation.

Note: The execution of this rule simply starts at the city specified in the field D_i and selects the closest city from the following cities of number specified in the field E_i and swaps the positions of the cities accordingly. This may be thought of a crude way of locally minimizing the distance of a sub-group of cities.

For rule 3:

D_i represents the city selected from which to locate nearest neighbors to.

E_i represents the number of nearest neighbors to find and swap positions with existing citiy neighbors.

F_i is ignored for this rule implementation.

For rule 4:

D_i represents the starting city for the group of cities to be reversed in order.

E_i represents the number of cities in the group to be re-ordered.

F_i is ignored for this rule implementation.

For rule 5:

D_i represents the first city in the selected group to re-order by distance based on roulette wheel selection.

E_i represents the number of cities to re-order.

F_i represents the position to initiate the random number generator for the roulette wheel spins.

Note: This rule is similar in nature to rule 3, however it allows for the selection of adjacent cities which are not nearest neighbors which is important to

avoid being trapped in a local minimum. The value specified by the field F_i is important in that it allows the rule to be executed in the exact way for each future generation.

5. The Test Problem Set

A series of test problems was generated randomly on a ten mile by ten mile rectilinear region. Problem size was varied with ten problems each at sizes of ten, twenty, fifty and one hundred cities. The *x* and *y* coordinates for each problem were stored in a data file which was subsequently read in to the various optimization algorithms. A solution for each set of ten problems was generated by simulated annealing, a traditional genetic algorithm and a rule based genetic algorithm. The results were then averaged for each solution method for each grouping of the same number of cities. While the ten city problem set was relatively easily solved, the difficulty increased significantly with the number of cities considered as expected. The algorithm performance results are summarized in Table 1.

From the above table several interesting observations can be made. First of all, the traditional genetic algorithm was not well suited for this class of problem. The ten city problem set was solved with the identical average number of path distance evaluations as for the simulated annealing algorithm. As the number of cities increased, however, the traditional genetic algorithm was simply incapable of locating a solution regardless of population size and number of generations allowed. This would indicate that the traditional genetic algorithm would have similar difficulty solving any routing or scheduling problem. The method of simulated annealing worked reliably on small scale problems, but it had difficulty locating the exact optimal solution. As expected, the number of path evaluations increased dramatically with problem size, and with additional experimentation in parameter selection, the results are likely to improve.

The interesting result is that the rule based genetic algorithm outperformed the simulated annealing technique both in the number of path evaluations re-

Table 1. Solution summary for various algorithms on traveling salesman problems of various size.

Number of Cities	Genetic Code	Genetic Code	Simulated Annealing	Simulated Annealing	Rule Based GA	Rule Based GA
	Average Function Evaluations	Times Best Solution Located	Average Function Evaluations	Times Best Solution Located	Average Function Evaluations	Times Best Solution Located
10	10800	10	10800	10	133	10
20	*	0	39600	10	2657	10
50	*	0	121500	4	76009	10
100	*	0	321000	1	228763	10

*indicates solution not achieved with reasonable number of evaluations.

quired as well as the ability to locate the best solution. No significant effort was made to establish the best operational parameters for the code. It should be noted that the rule based code utilized an initial population based on randomly generated paths. It is difficult to compare results between the rules based genetic algorithm and the simulated annealing algorithm as they arrived at different solutions for many of the test problems. In general the speed of solution may be traced directly to the number of path evaluations which are plotted in Figure 2.

The number of function evaluations required for the rule based genetic algorithm is significantly below that required by the simulated annealing algorithm. Both algorithms demonstrate a significant increase in required function evaluations as problem size increases which points out the difficulty in solving this class of problems as the scale increases. Exact numbers of evaluations may vary with input parameter selection and as such exact comparisons should be made with caution. It definitely can be stated, however, that the performance of the rule based genetic algorithm is as good or superior to the simulated annealing technique and orders of magnitude better than the performance of a traditional genetic algorithm.

The solution for one of the problems of each size is pictured in the following Figures 3-6.

From Figures 3-6, it can be seen that the solutions generated are reasonable paths to minimize the distance traveled. It can also be seen how the difficulty of the problem increases with the number of cities considered.

The results on these randomly generated problem sets demonstrate the potential of a rule based genetic approach. It was the only algorithm of those tested which was capable of consistently generating global solutions to the test problems. The other algorithms could generate local optimal solutions that were within a few percent of the global optimal solution. This demonstrates that even

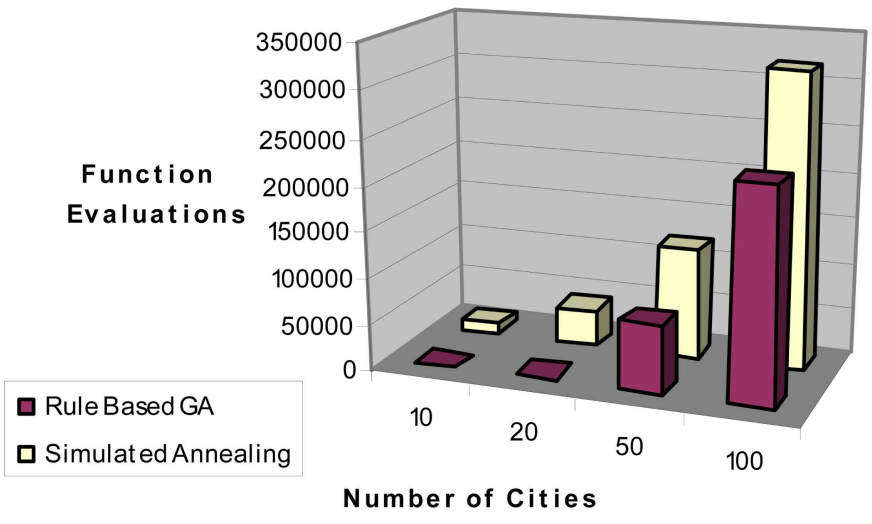

Figure 2. Comparison of number of the average number of function evaluations required for solution between algorithms.

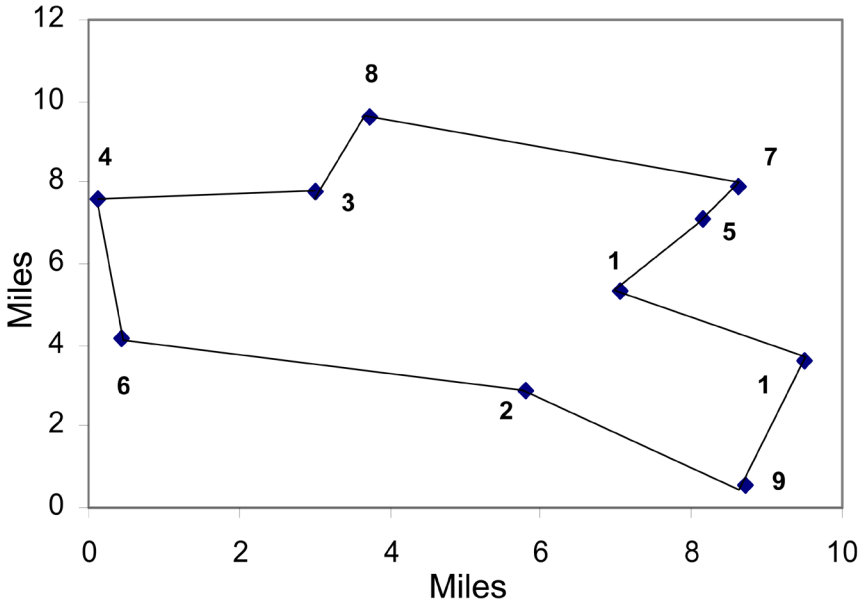

Figure 3. Solution path for a ten city problem.

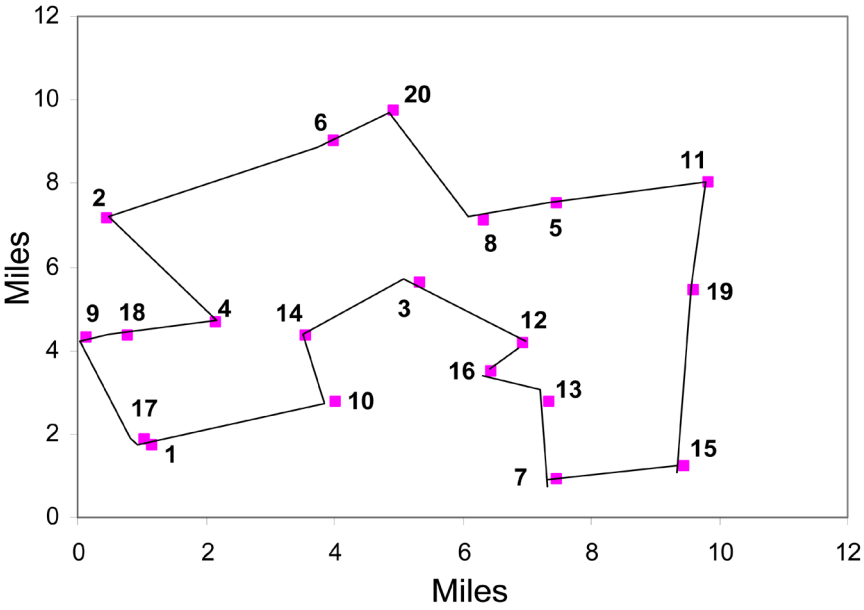

Figure 4. Solution path for a twenty city problem.

though the simulated annealing approach was designed to find global minima, this is not always the case. The other point of note is in the number of function evaluations required to generate the solution. The rule based algorithm was more efficient, although the efficiency dropped off as the problem size increased. Even for the one hundred city test group, however, the number of function evaluations was considerably lower. Thus, the rule based approach was more robust in relation to locating the global solution as well as more efficient in the solution time required.

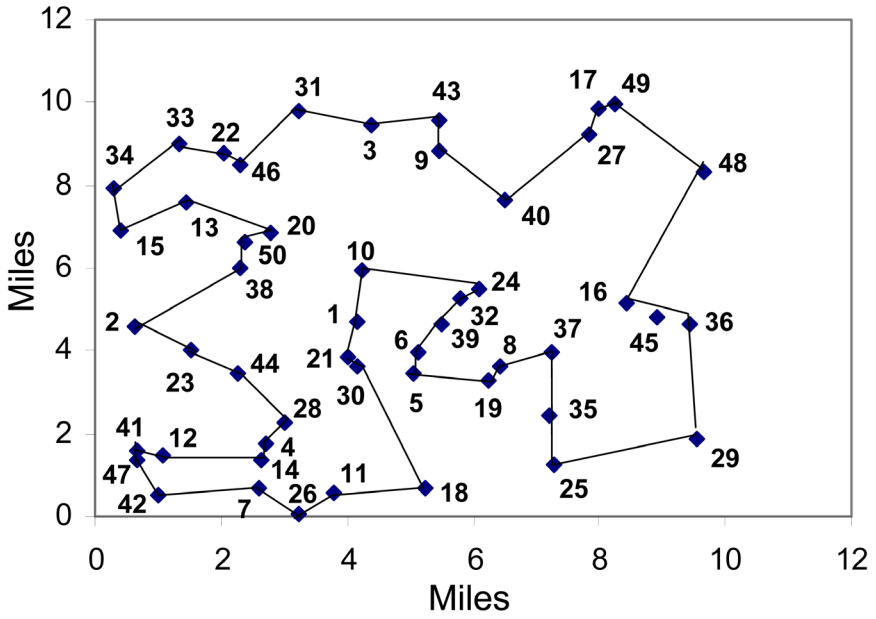

Figure 5. Solution path for a fifty city problem.

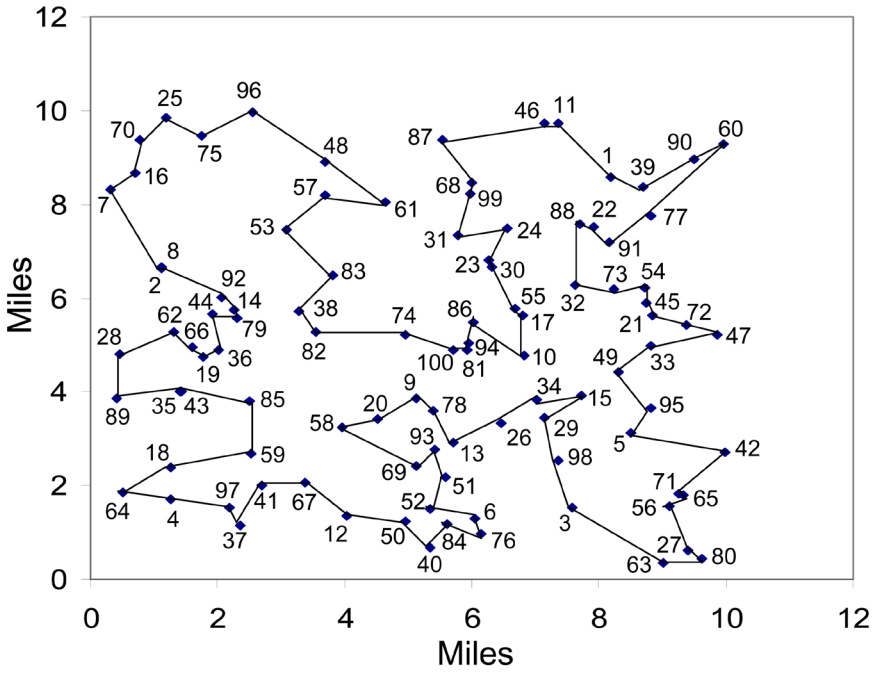

Figure 6. Solution path for a one hundred city problem.

6. Evaluation of Rule Selection

As was stated previously, no significant effort was put into selecting the rule set for this problem for the rule based genetic algorithm. The performance of the algorithm could potentially be enhanced significantly by careful evaluation of which rules factored into the solution process and then modifying rules or selecting new rules based on this information. As the genetic algorithm is capable

of maintaining a solution history, an evaluation of the implemented rules may be made following single or multiple problem solutions. The following Figures 7-10 present the percentage of the time a successful new path was located with each of the rules, averaged over the ten problems in each set. A successful new path is simply a path which is better than the current best path in the population.

From Figures 7-10, several observations may be made. First of all, each of the rules was utilized in the solution process of the rule based genetic algorithm. Each of the five rules is fairly equally utilized for the ten and twenty city problem sets. As the problem size increases, it appears that rules 2, 3 and 4 begin to be-

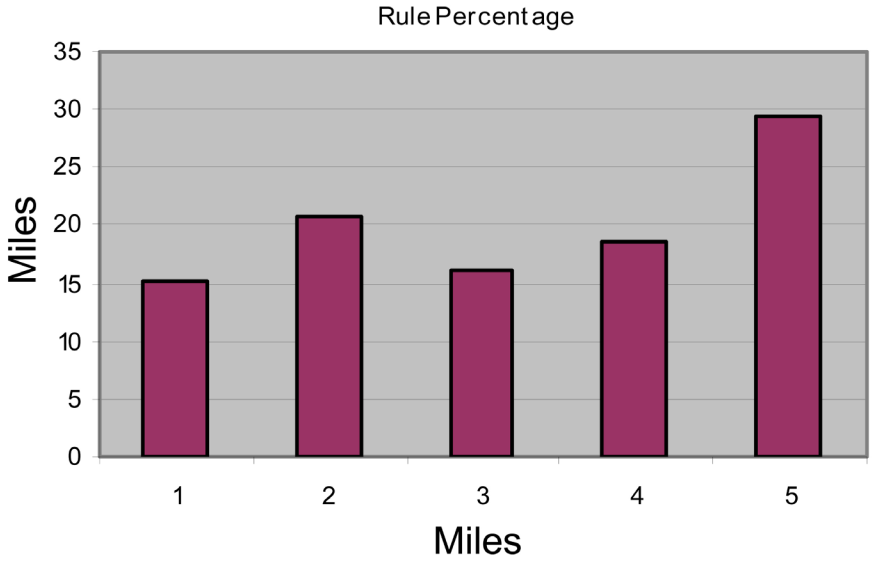

Figure 7. Successful rule percentages for ten city problem.

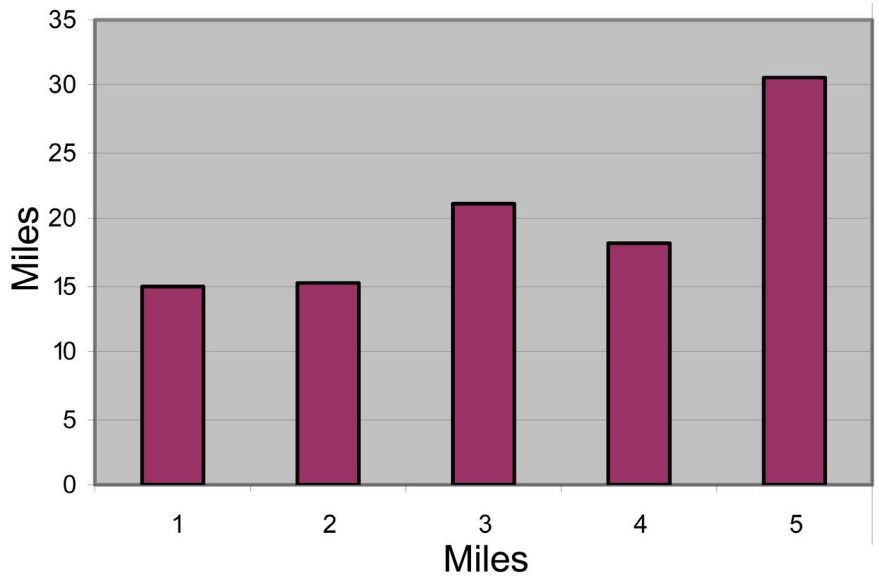

Figure 8. Successful rule percentages for twenty city problem.

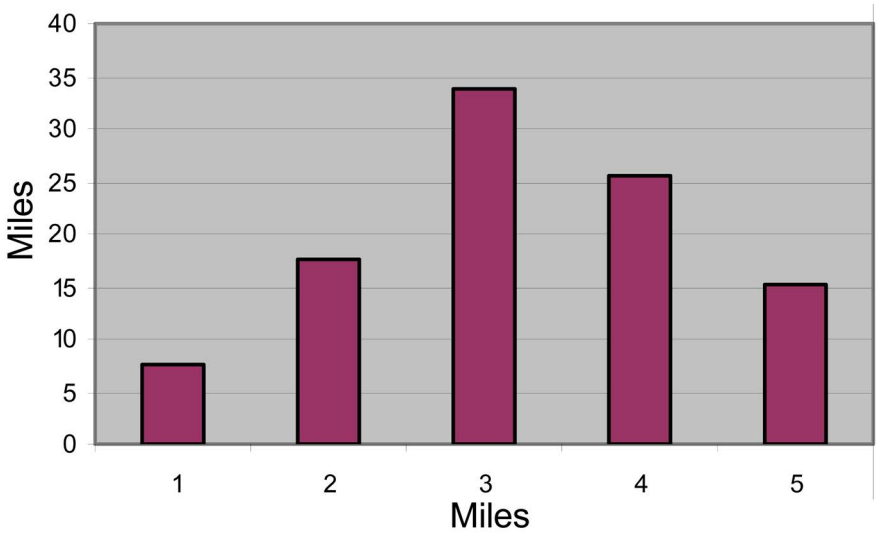

Figure 9. Successful rule percentages for fifty city problem.

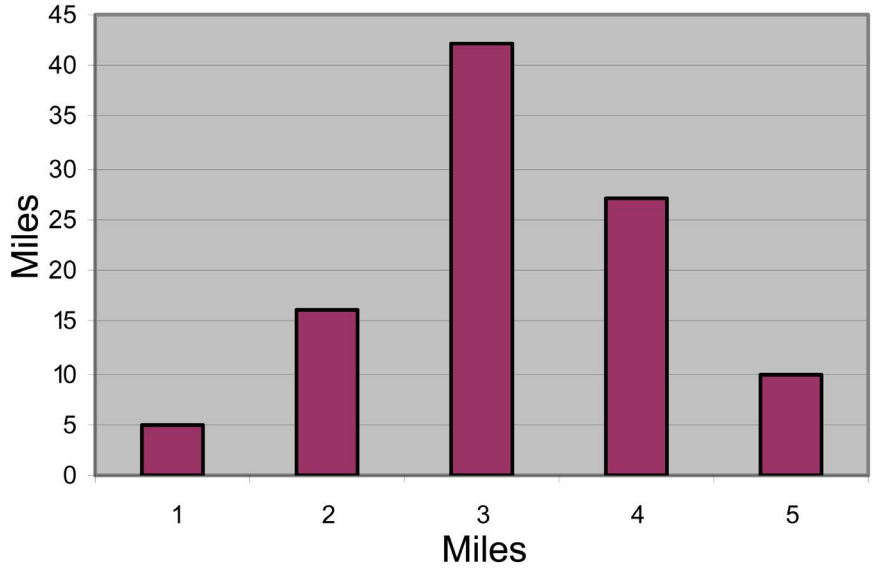

Figure 10. Successful rule percentages for one hundred city problem.

become more dominant. This points out the fact that as the problem size increases, the type of rules implemented may need some modification to maintain an efficient solution. The advantage is that this information is available and can help improve the performance of the rule based algorithm on a particular problem class. Specific rules for the traveling salesman problem are difficult as the minimum distance path leaves little room for interpretation. The rule set for a scheduling or delivery route planning problem would have obvious rules which could be productively implemented. For example, in a manufacturing scheduling problem with machine setups, one rule might be to try and group jobs which require little or no set-up time on a specific machine. Rules such as this will guide the solution through local minima, to a representative, global solution. It

certainly makes sense to include as much problem specific information as possible for a particular problem class. On the other hand, one must guard against generating a greedy algorithm which quickly locates a local, rather than a global solution, unless a local solution is all that is desired. Local solutions can be avoided by maintaining some rules which can perform generic alteration in a design string which will help maintain the global character of the genetic solution process.

In order to demonstrate the robust nature of the rule based evolutionary approach, an additional 15 problems were solved. These problems were selected from the test problem set maintained by the University of Heidelberg and represent traveling salesman problems varying in the number of cities. The best known solution for each of the problems is also recorded. Fifteen problems ranging from fifty to 200 cities were selected and solutions were generated. The results generated by the rule based algorithm are documented in Table 2. From this table it can be seen that the rule based algorithm located the global solution in every case. This is further confirmation that the algorithm is robust and is able to locate a global optimum for most cases.

7. Summary and Conclusions

The framework for a general rule based, evolutionary environment is presented which can be utilized as an intelligent platform for the investigation of heuristic approaches for solving the traveling salesman and other related decision support problems. A set of five simple rules was embedded into the encoding of the ge-

Table 2. Results on TSPLIB problems.

Problem	Number of Cities	Best known Solution	Rule Based GA Result
Eil 51	51	426	426
Berlin 52	52	7542	7542
Eil 76	76	538	538
Rat 99	99	1211	1211
KroA 100	100	21282	21282
KroB 100	100	22141	22141
KroC 100	100	20749	20749
Eil 101	101	538	538
Pr 107	107	44303	44303
Bier 127	127	118282	118282
Ch 130	130	6110	6110
Ch 150	150	6528	6528
D198	198	15780	15780
Pr 226	226	80369	80369
A280	280	2579	2579

netic algorithm for the evaluation of the feasibility of the approach. The rule based algorithm was tested on sets of ten problems with randomly generated city locations for problem sizes including ten, twenty, fifty and one hundred cities. For comparison purposes, a traditional genetic algorithm and a simple simulated annealing algorithm were utilized to solve the same problem sets. The results demonstrate the potential of the rule based genetic approach. None of the tested algorithms were tuned or tailored for efficiency on the problem set. The rule based approach was able to consistently locate better quality solutions with less computational effort that the other two algorithms tested. Additionally, fifteen problems from a recognized test set of traveling salesman problems were solved and compared to the best known solutions. In every case the new algorithm located the global optimum.

The post evaluation of the results from the test problem sets point out which of the rules applied had the most positive impact on the search process. This analysis allows for the replacement of non-effective rules and for refinement of rules to increase their impact of the process. The effectiveness of individual rules was shown to be a function of problem size which is an indication that the rule set may have to be more sophisticated in order to solve problems involving one thousand cities or more. Additional tracking of rule combinations that proved to be effective could be easily accomplished. The concept is to provide an open computational platform which can be improved in performance over time to become increasingly effective in solving a particular problem class. A wide range of heuristic options may be provided initially and paired down over time.

The proposed algorithm executes a population of rules on a small population of potential city routes. This allows for large problems to be executed with a limited population size that need not grow significantly as the problem size increases. Currently, the initial city route population is generated randomly, but the possibility of utilizing a two phase approach is real option. This approach would cycle between a conventional genetic approach operating upon the specified city order encoding and the rule based approach improving the members of the phase one population. Additional rules such as the so called double bridge move proposed by Martin, Otto and Felton [20] may be easily incorporated. The existing framework, even with the simplistic rule set employed provides the basis for continued development and application to larger sets of problems.

References

[1] Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M. (1954) Solution of a Large-Scale Traveling Salesman Problem. *Operations Research*, **2**, 393. https://doi.org/10.1287/opre.2.4.393

[2] Balas, E. and Toth, P. (1985) Branch and Bound Methods. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B., Eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, New York, 80-147.

[3] Lo, C.C. and Hus, C.C. (1998) Annealing Framework with Learning Memory. *IEEE*

Transactions on System, *Man*, *Cybernetics*, *Part A*: *Systems and Humans*, **28**, 1-13.

[4] Pepper, J.W., Golden, B.L. and Wasil, E.A. (2002) Solving the Traveling Salesman Problem with Annealing-Based Heuristics: A Computational Study. *IEEE Transactions on System*, *Man*, *Cybernetics*, *Part A*: *Systems and Humans*, **32**, 72-77. https://doi.org/10.1109/3468.995530

[5] Glover, F. Tabu Search—Parts I and II. *ORSA J. Computing*, Vol. 1, 190-206, 1989 and Vol. 2, 4-32, 1990.

[6] Tsai, C.F. and Tsai, C.W. (2002) A New Approach for Solving Large Traveling Salesman Problem Using Evolutionary Ant Rules. *Neural Networks*, 2002 *IJCNN* '02 *Proceedings of the* 2002 *International Joint Conference*, **2**, 1540-1545.

[7] Modares, A., Somhom, S. and Enkawa, T. (1999) A Self-organizing Approach for Multiple Traveling Salesman and Vehicle Routing Problems. *International Transactions in Operations Research*, **5**, 591-606. https://doi.org/10.1111/j.1475-3995.1999.tb00175.x

[8] Julstrom, B.A. (1995) Very Greedy Crossover in a Genetic Algorithm for the Traveling Salesman Problem. SAIC '95: Proceedings of the 1995 ACM Symposium on Applied Computing, Nashville, Tennessee, 26-28 February 1995, 324-328. https://doi.org/10.1145/315891.316009

[9] Wang, L.Y., Zhang, J. and Li, H. (2007) An Improved Genetic Algorithm for TSP. *Machine Learning and Cybernetics*, 2007 *International Conference*, 2, 925-928. https://doi.org/10.1109/icmlc.2007.4370274

[10] Kindervater, G.A.P. and Lenstra, J.K. (1985) Parallel Algorithms. In: O'hEigeartaigh, M., Lenstra, J.K. and Rinnooy, A.G., Eds., *Combinatorial Optimization*: *Annotated Bibliographies*, Wiley, Chichester, 106-128.

[11] Osman, I. and Kelly, J. (1996) Meta-Heuristics: An Overview. In: Osman, I. and Kelly, J., Eds., *Meta-Heuristics*: *Theory and Applications*, Kluwer, Boston, 1-21.

[12] Katayama, K., Hirabayashi, H. and Narihisa, H. (1998) Performance Analysis of a New Genetic Crossover for the Traveling Salesman Problem. *IEICE Transactions on Fundamentals of Electronics*, *Communications and Computer Sciences*, **E81-A**, 738-750.

[13] Applegate, D.L., Bixby, R.E., Chvatal, V. and Cook, W.J. (2007) The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton.

[14] Agharghor, A., Riffi, M.E. and Chebihi, F. (2016) A Memetic Hunting Search Algorithm for the Traveling Salesman Problem. 2016 4*th IEEE International Colloquium on Information Science and Technology*, Tangier, 206-209. https://doi.org/10.1109/cist.2016.7805043

[15] Kaur, D. and Murugappan, M.M. (2008) A Genetic Algorithm Balancing Exploration and Exploitation for the Traveling Salesman Problem. *Annual Meeting of the North American Fuzzy Information Processing Society Fuzzy Information Processing Society*, Annual Meeting of the North American, 1-6 May 2008.

[16] Reinelt, G. TSBLIB 95, Universitat Heidelberg, Institut fur Angewandte Mathematik.

[17] Metropolis, N., Rosenbluth, A., Teller, M. and Teller, E.J. (1953) Equations of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, **21**, 1087-1092. https://doi.org/10.1063/1.1699114

[18] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1986) Numerical Recipes. Cambridge University Press, New York.

[19] Lin, S. (1965) Computer Solution of the Traveling Salesman Problem. *The Bell System Technical Journal*, **44**, 2245-2269. https://doi.org/10.1002/j.1538-7305.1965.tb04146.x

[20] Martin, O., Otto, S.W. and Felten, W. (1992) Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics. *Operations Research Letters*, **11**, 219-224.