Design and Application of Genetic Algorithms for the Multiple Traveling Salesperson Assignment Problem

by

Arthur E. Carter

Dissertation submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Business

APPROVED

April 21, 2003
Blacksburg, Virginia

Keywords: Multiple Traveling Salesperson Problem, Genetic Algorithms, Artificial Intelligence

Design and Application of Genetic Algorithms for the Multiple Traveling Salesperson Assignment Problem

by

Arthur E. Carter

Abstract

The multiple traveling salesmen problem (MTSP) is an extension of the traveling salesman problem with many production and scheduling applications. The TSP has been well studied including methods of solving the problem with genetic algorithms. The MTSP has also been studied and solved with GAs in the form of the vehicle-scheduling problem. This work presents a new modeling methodology for setting up the MTSP to be solved using a GA. The advantages of the new model are compared to existing models both mathematically and experimentally. The model is also used to model and solve a multi line production problem in a spreadsheet environment. The new model proves itself to be an effective method to model the MTSP for solving with GAs. The concept of the MTSP is then used to model and solve with a GA the use of one salesman make many tours to visit all the cities instead of using one continuous trip to visit all the cities. While this problem uses only one salesman, it can be modeled as a MTSP and has many applications for people who must visit many cities on a number of short trips. The method used effectively creates a schedule while considering all required constraints.

DEDICATION

I dedicate this work to my wife, Tracy. Without her support, encouragement, understanding, patience and faith in my abilities, this work, and my goal of obtaining a Ph.D., would never have been possible.

ACKNOWLEDGEMENTS

I'd like to thank the members of my committee for the support I received. I would like to especially thank Dr. Cliff Ragsdale for taking on the job of chairing my committee. I know that there was a great deal of effort, patience and dedication on his part during the several years that have gone into this work. I would also like to thank Dr. Loren Rees for the advice and guidance he provided, especially early in the process, which made it possible for me to get where I am in the time frame I chose. I also thank Dr. Lance Matheson, Dr. Debbie Cook and Dr. Evelyn Brown for serving on my committee. I know that it required time and effort on their part with the only return being the knowledge that they helped a student achieve his academic goals.

I would also like to thank Dr. Ernie Houck and Dr. Bernard Taylor. Their assistance and support made it possible for me to work at Va Tech on my Ph.D.

TABLE OF CONTENTS

CHAPTER 1: Introduction and Literature Review 1

Introduction 2
Approaches to Solving the TSP 3
Genetic Algorithms and the Multiple Traveling Salesperson Problem 4
A Production Scheduling Problem Modeled as a Multiple Salesperson TSP 8
One Salesperson with Multiple Tours 10
Future Research and Conclusions 11

CHAPTER 2: A New Approach to Solving the Multiple Traveling Salesperson Problem Using Genetic Algorithms 12

Introduction 13
Literature Review 14
Overview of GAs 14
Chromosome Representations for the MTSP 16
- The Two Chromosome Method 16
- The One Chromosome Method 17
- The Two Part Chromosome Method 17

Comparison of Solution Spaces 18
Computational Testing Methodology 20
- GA Issues 20
  - Two Chromosome Technique 21
  - One Chromosome Technique 21
  - Two-Part Chromosome Technique 21
- Computational Results: Minimum Total Distance 21
- Computational Results: Minimum Longest Tour 24

Conclusions 28

CHAPTER 3: A Production Scheduling Problem Modeled as a Multiple Salesperson TSP 29

Introduction 30
Advertising Inserts and the Newspaper Industry 31
Overview of Genetic Algorithms 34
Using a GA to Optimize Insert Scheduling 34
Example Problem 36
The Model 37
Computational Testing and Results 39
Conclusions 40

CHAPTER 4: Quality Inspection Scheduling for Multi-Unit Service Enterprises 41

Introduction 42
Overview of the TQIP 43
Modeling the TQIP 44
Example Problem 46
Solution Framework 48
Overview of Genetic Algorithms 50
A Custom GA For The TQIP 51
    Chromosome Design 51
    Evolutionary Operators 52
    Fitness Evaluation 54
    Initial Population Generation 54
Methodology 55
Computational Testing and Results 56
    Sensitivity Analysis of Penalties 58
    Longitudinal Analysis 60
    The Results of Using Best-Trip 61
Benefits of Proposed Methodology 63
Conclusions 64

CHAPTER 5: Summary and Directions for Future Research 65

Summary 66
Future Research 66
    Vehicle Secluding Problem 66
        Time Windows 66
        Pick Up/Drop off precedents 67
        Minimum Trucks/Minimum Distance 67
    DSS Development for Salesperson/Inspectors 67
    New or Improved Operators for the MTSP 67
Conclusions 67

REFERENCES 69

APPENDIX A 74

CURRICULUM VITAE 77

LIST OF TABLES

Table 2-1: Computational Tests 20
Table 4-1: Sample of Schedules Created Using Date and Sequence Priority Methods 56

LIST OF FIGURES

Figure 1-1: TSP Solution with Two Subtours 3

Figure 1-2: Example of a Two Chromosome Representation of a 15 City MTSP with 4 Salespersons 6

Figure 1-3: Example of a One Chromosome Representation of a 15 City MTSP with 4 Salespersons 6

Figure 1-4: Example a Two-Part Chromosome Representation of a 15 City MTSP with 4 Salespersons 7

Figure 2-1: Example of a Two Chromosome Representation of a 15 City MTSP with 4 Salespersons 16

Figure 2-2: Example of a One Chromosome Representation of a 15 City MTSP with 4 Salespersons 17

Figure 2-3: Example a Two-Part Chromosome Representation of a 15 City MTSP with 4 Salespersons 18

Figure 2-4: Comparison of Chromosome Solution Spaces for a 15 City MTSP with m Salespersons 19

Figure 2-5: Averages for 51 City Problem Minimizing Total Distance 22

Figure 2-6: 51 City Problem Using Total Distance as Fitness Function 22

Figure 2-7: Average for 100 City Problem Minimizing Total Distance 22

Figure 2-8: 100 City Problem Using Total Distance as Fitness Function 23

Figure 2-9: Average for 150 City Problem Minimizing Total Distance 23

Figure 2-10: 150 City Problem Using Total Distance as Fitness Function 24

Figure 2-11:	Averages for 51 City Problem Minimizing the Longest Tour	25
Figure 2-12:	51 City Problem Using Longest Route as Fitness Function	25
Figure 2-13:	Averages for 100 City Problem Minimizing the Longest Tour	26
Figure 2-14:	100 City Problem Using Longest Route as Fitness Function	26
Figure 2-15:	Averages for 150 City Problem Minimizing the Longest Tour	27
Figure 2-16:	150 City Problem Using Longest Route as Fitness Function	27
Figure 3-1:	Newspaper Production Flow Diagram	32
Figure 3-2: on 4 Lines	Example Chromosome Representing a Solution for 15 Zones	35
Figure 3-3:	Sample spreadsheet for the insert scheduling problem	38
Figure 3-4:	Estimated Processing Time Produced By GA Solutions	39
Figure 4-1:	Unit Inspection Window Example	48
Figure 4-2:	Model for TQIP Instance	49
Figure 4-3:	Summary Computational Results	57
Figure 4-4:	Breakdown of Fitness Function Values	57
Figure 4-5:	Sensitivity Analysis Results - Minimizing Window Violations	58
Figure 4-6:	Sensitivity Analysis Results - Minimizing Long Trips	59
Figure 4-7:	Sensitivity Analysis Results - Minimizing Distance	59
Figure 4-8:	Simulation Results	61
Figure 4-9:	Graphical Display of Sensitivity Analysis	62

Chapter 1

Introduction and Literature Review

INTRODUCTION

The traveling salesperson problem (TSP) is a classic model for various production and scheduling problems. Many production and scheduling problems ultimately can be reduced to the simple concept that there is a salesperson who must travel from city to city (visiting each city exactly once) and wishes to minimize the total distance traveled during his tour of all *n* cities. Obtaining a solution to the problem of a salesperson visiting *n* cities while minimizing the total distance traveled is one of the most studied combinatorial optimization problems. While there are variations of the TSP, the Euclidean TSP is NP-hard (Schmitt & Amini, 1998; Falkenauer, 1998). The interest in this particular type of problem is due to how common the problem is and how difficult the problem is to solve when *n* becomes sufficiently large.

The TSP can be formulated as an integer linear programming model as follows:

$$\text{MIN: } \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}X_{ij} \qquad (1)$$

$$\text{S.T. } \sum_{i=1}^{n} X_{ij} = 1, \qquad j = 1,\ldots, n \qquad (2a)$$

$$\sum_{j=1}^{n} X_{ij} = 1, \qquad i = 1,\ldots, n \qquad (2b)$$

$$\{(i,j) \mid i,j = 2, \ldots, n;\ X_{ij} = 1\} \text{ does not contain subtours,} \qquad (3)$$

$$X_{ij} \in \{0,1\} \quad \text{for all } i,j = 1,\ldots, n \qquad (4)$$

where c_{ij} is the distance from city *i* to city *j*. If $c_{ij} = c_{ji}$ for all *i* and *j* then the problem is symmetric, otherwise it is asymmetric. The binary variable X_{ij} equals one if the route from city *i* to *j* is in the solution and zero otherwise. The objective in (1) is to minimize the total distance traveled. Constraints (2a) and (2b) ensure, respectively, that each city is entered and exited exactly once. Unfortunately, it is possible for feasible solutions to constraints (2a) and (2b) to produce subtours where, instead of having one continuous path connecting all of the cities, there are multiple smaller paths (or subtours) that include all of the cities. For example, Figure 1-1 shows a TSP with two subtours that satisfy constraints (2a) and (2b) but not constraint (3).

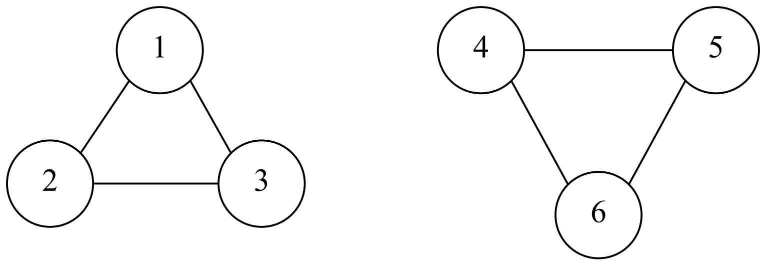


Figure 1-1
TSP Solution with Two Subtours

Constraint (3) explicitly prevents subtours and can be written in a number of ways depending on the formulation of the problem (Gouveia & Pires, 2001). This constraint makes the problem difficult to solve for linear programming-based solution techniques because, for an n-city problem, constraint (3) results in $2^n - 2$ subtour constraints for the problem (Garfinkel & Nemhauser, 1972).

APPROACHES TO SOLVING THE TSP

Research into solving the TSP has lead to a number of different methods of obtaining either the optimal or a good solution. Several works have been produced that provide background on TSPs and the methodology to solve them (Lawler, Lenstra, Rinnooy Kan, & Shimoys, 1985). The methods used to solve the TSP include classic methodologies like linear programming (Wong, 1980) and branch-and-bound (Little, Murty, Sweeney and Karel, 1963; Bellmore & Malone, 1971) and artificial intelligence methods like neural networks (Shirrish, Nigel & Kabuka, 1993) tabu search (Glover, 1990) and genetic algorithms (Goldberg & Lingle, 1985). An ideal solution method would solve every TSP problem to optimality, but this is not practical in most large problems. While advances have been made in solving the TSP, those advances have come at the cost of more complicated computer code. The complexity involves not only the length of the code, but the required nesting and data structures (Chatterjee, Carrera & Lynch, 1996).

Genetic algorithms (GAs) are a relatively recently developed optimization technique that can be applied to TSPs. GAs were developed by John Holland and were not originally developed for use on constrained problems like the TSP (Holland, 1975). Later, GAs were adapted to constrained optimization problems including order-based problems like TSP (Goldberg & Lingle, 1985; Jog, Suh and Van Gucht, 1989; Whitley, Starkweather & Fuquay, 1989). The development of effective GA operators for TSPs has lead to a great deal of interest and research to improve GAs' performance for this type of problem (Poon & Carter, 1995; Qu & Sun, 1999; Katayma & Sadamoto, 2000). Several reviews of solving TSPs with GAs have been published. These works include a comprehensive look at the operators that have been developed and the issues associated with the various operators (Potvin, 1996; Schmitt & Amini 1998; Schaffer, Eshelman, & Offutt, 1991).

GENETIC ALGORITHMS AND THE MULTIPLE TRAVELING SALESPERSON PROBLEM

The work on TSPs has focused on the idea of a *single* salesperson traveling in a continuous trip visiting all *n* cities exactly once and returning to the starting point. There has been some work done on the idea of using multiple salespersons to visit all of the cities exactly once. The multiple TSP work has focused on vehicle scheduling problems with the goal of minimizing the total distance traveled by all the trucks (salespersons) or minimizing the total number of trucks required (Malmborg, 1996; Park, 2001). The goal of balancing the workload (or distance traveled) among the multiple salespersons that are being used to visit the required cities is a problem that has been left largely unexamined. A TSP with a single salesperson problem is very difficult. Using multiple salespersons makes the problem even harder due to the increased number of ways to visit each city once. While the multiple TSP can be modeled as a single TSP, the result is effectively a longer tour that increases the solution space of the problem.

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are evaluated according to a fitness (or objective) function, with the fittest surviving and less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem (Bergy & Ragsdale, 1999). The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes, or until no significant improvement occurs in the population for a given number of iterations.

The key to finding a good solution using a GA lies in developing a good model of the problem. A good GA model should reduce or eliminate redundant chromosomes from the population and provide GA operators that effectively improve the population of solutions. The chromosome must accurately represent the problem and allow the GA operators to work effectively on the chromosomes to generate better solutions as the iterative process goes on. An example of a GA model designed for the efficient solution for a specific problem is the grouping genetic algorithm developed by Falkenauer (1998). Falkenauer's grouping genetic algorithm increases the effectiveness of GAs on grouping problems (bin packing, workshop layout and graph coloring) due to the modeling of the problem in a way that reduces the redundancy of having the same solution being encoded in multiple different ways. The chromosome must also be of a nature that allows the fitness of the solution being represented by the chromosome to be calculated easily. This facilitates comparison of the various chromosomes to determine which chromosomes are better and should remain in the population of solutions.

Solving the TSP using GAs has generated a great deal of research that has focused on how best to perform the action of "evolving" an optimal (or good) solution to the problem. The operators used in the GA are key in the ability of a GA to find a good solution to the problem. The TSP is unique in that it requires the solution to have each city visited once and only once. This requires the operators to perform crossover and generate the new child solutions while maintaining a feasible solution (one where each city is visited once and only once).

A number of different crossover methods have been proposed to solve the TSP using a GA. Some of the most commonly used operators include: Order Crossover (Goldberg, 1985; Davis, 1985; Starkweather, Whitley, Whitley, & Mathial, 1991; Oliver, Smith & Holland, 1987), Partially Mapped Crossover (Goldberg, 1985; Starkweather, et al, 1991; Goldberg and Lingle, 1985; Oliver, Smith & Holland, 1987), Cycle Crossover (Goldberg, 1985; Starkweather, et al, 1991; Oliver, Smith & Holland, 1987) and Asexual Crossover (Chatterjee, et al, 1996; Schmitt and Amini, 1998; Fox & McMahon, 1991). Other TSP operators that have been proposed include utilizing a matrix chromosome representation (Poon and Carter, 1995; Qu and Sun, 1999), hybrid operators (Laiw, 2000), simple crossover (Knosala & Wal, 2001) ordered crossover #2 (Syswerda, 1991), moon crossover (Moon, Kim, Choi & Seo, 2002) and position based crossover (Syswerda, 1991).

Multiple traveling salespersons problems (MTSP) are similar to TSPs except that there is more than one salesperson available to travel to the *n* cities (though each city must still be visited exactly once). MTSPs have been modeled and solved with GAs by those solving vehicle scheduling problems (VSP). The VSP consists of scheduling a fleet of vehicles to visit *n* cities with each city being visited by one and only one truck. The VSP faces constraints on the number of cities each truck can visit due to the capacity of each truck available and the size of the load to be picked up at each city. In some cases, the cities must be visited between specific times called time windows. A variety of objectives can be considered: minimize total distance, minimize number of trucks required or minimize lateness (if time windows are used). These issues lead to a number of different possible configurations for the VSP: VSP with/without time windows, VSP with heterogeneous/homogeneous truck capacities, and VSP for minimum distance/minimum truck requirements.

If modeled properly the MTSP can be solved using any of the GA operators developed for the TSP. Researchers have proposed two methods of modeling the MTSP such that a TSP operator can be used. One method uses two chromosomes to model the MTSP. The first chromosome provides a permutation of the cities and the second chromosome assigns a salesperson to the city in the corresponding position of the first chromosome (Malmborg, 1996; Park, 2001). This method requires two chromosomes of length *n* to represent a solution. For example see Figure 1-2.

Cities

2	5	14	6	1	11	8	13	4	10	3	12	15	9	7

Salespersons

2	1	1	3	4	3	2	4	4	1	3	2	1	2	3

Figure 1-2
Example of a Two Chromosome Representation of a 15 City MTSP with 4 Salespersons.

In the example in Figure 1-2, cities 2, 8, 12 and 9 would be visited by salesperson 2 in that order. Cities 5, 14, 10 and 15 would be visited by salesperson 1 and likewise for the remaining cities assigned to salespersons 3 and 4. If *m* represents the number of salespersons, using this modeling scheme, there are $n!(m^n)$ possible solutions to the problem, many of which are redundant.

The second method is to model the MTSP as a single TSP with one chromosome and lengthening the chromosome by one position for each salesperson above one that is going to be used to cover the routes. In this model the extra positions in the chromosome are filled with dummy cities that represent the change from one salesperson to the next. For example see Figure 1-3.

Cities

2	5	14	6	-2	1	11	8	13	-3	4	10	1	-1	12	15	9	7

Figure 1-3
Example of a One Chromosome Representation of a 15 City MTSP with 4 Salespersons.

In the example in Figure 1-3 the negative numbers are used as the dummy cities to indicate the change from one salesperson to the next. So the first salesperson would visit cities 2, 5, 14 and 6 in that order. The second salesperson would visit cities 1, 11, 8 and 13 in that order, and so on for salespersons 3 and 4. This makes the chromosome $n + m - 1$ in length where *m* is the number of salespersons (Tang, Liu, Rong & Yang, 2000). Using this modeling scheme, there are $(n + m - 1)!$ possible solutions to this problem, many of which are redundant.

This research presents a new method for using a GA to solve the MTSP using a two-part chromosome to model the MTSP. The idea of using a two-part chromosome for the MTSP is similar to the two-part chromosome of the grouping genetic algorithm (Falkenauer, 1998). The grouping genetic algorithm was developed for solving grouping problems that require the grouping of some items in such a manner as to maximize (or minimize) some fitness criteria. The problem occurs in such problems as bin packing, workshop layout and color graphing (Falkenauer, 1998). Grouping problems do not have requirements for specific ordering like the MTSP. Falkenauer's grouping genetic algorithm uses a variable length second part which is not suitable for modeling MTSP, but which provide idea for creating a two-part chromosome that can model the MTSP.

The new modeling technique developed for the MTSP uses a two-part chromosome that also reduces redundancy of the same solution being represented in different ways. For the MTSP, the first part of the chromosome is a permutation of the *n* cities. The second part of the chromosome is of length *m* and represents the number of cities assigned to each of the *m* salesperson. The sum of the values assigned to all *m* salespersons must sum to the number of cities to be visited (*n*). See Figure 1-4.

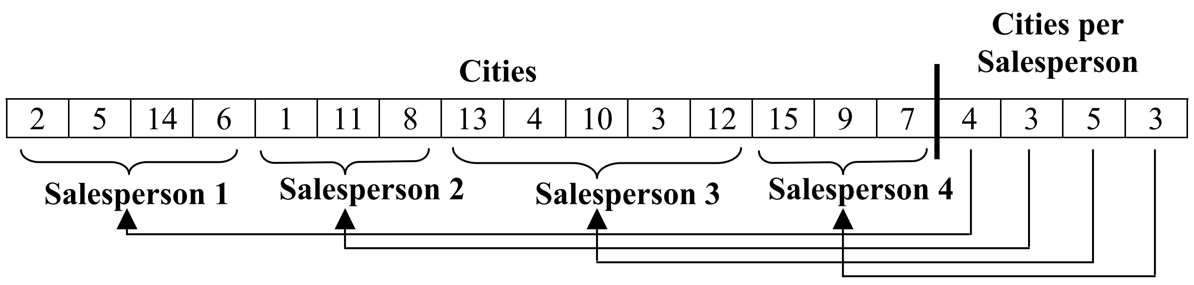


Figure 1-4
Example a Two-Part Chromosome Representation of a 15 City MTSP with 4 Salespersons.

In Figure 1-4, salesperson 1 would visit cities 2, 5, 14 and 6 in that order, salesperson 2 would visit cities 1, 11 and 8 in that order, and so on for salespersons 3 and 4. Using the new two-chromosome method reduces the size of the search space due to the efficiency of this modeling technique. A mathematical comparison of these three techniques and empirical test results are the focus of chapter 2.

A PRODUCTION SCHEDULING PROBLEM MODELED AS A MULTIPLE SALESPERSON TSP

The idea of multiple salespersons being used to complete a visit to each city is an applicable model for many scheduling and production scenarios. One salesperson visiting multiple cities is often used to model a production line which must process a certain number of jobs of varying length with varying set up times (or costs) between any two jobs. The fact that each job may require different amounts of time to process does not affect the solution to the problem because the time to process the job is constant whether the job is first, last or anywhere in between. What varies is the time to change the machine setup between jobs and that time can be different between any two jobs. Also, since the initial machine setup and return to the original setup can be done during non-production time, they do not need to be included in the tour. So unlike the classic TSP, the production scheduling problem does not have to return to the original origin city (or does so implicitly). This results in the objective function of the model becoming:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}X_{ij} + \sum_{i=1}^{n} l_i \tag{5}$$

Where:

l_i = the length of time required to process job i

c_{ij} = the time required to change from job i to job j

X_{ij} is a binary variable which is 1 if job j follows job i, otherwise its 0

The term for the sum of l_i is the same for every solution to the problem. Therefore the goal becomes ordering the jobs such that the time required to change setups between the jobs is minimized.

If multiple production lines are being used to produce these same jobs then it becomes a TSP with multiple salespersons. While this can be handled by assigning each line a set of jobs and then optimizing each line (thus turning the problem into multiple single salesperson TSPs) that will not, in general, provide the best solution. An additional consideration is that each line may run at different speeds. When this is added to the problem, the time to process each load may vary based on the nature of the job and the specific line to which the job is assigned. When the goal is to balance the loads between the lines such that the longest running time on any line is minimized, the set of jobs assigned to each line must be treated as a decision variable. The total time required to process the jobs assigned to a given line is computed as:

$$t_j = \sum_{i=1}^{n_i} (c_{ij} + l_{ij}) \qquad (6)$$

where:

l_{ij} = time required to process load i on line j

n_j = number of loads assigned to line j

c_{ij} = time required to changeover to the i^{th} load on line j

The objective function for the MTSP with a goal of minimizing the maximum production time on m lines would be represented as follows:

$$\text{MIN:} \quad \mathbf{MAX}(t_1, t_2, \ldots, t_m) \qquad (7)$$

The production line scheduling problem can be effectively modeled as a TSP and has been extensively studied (Wang, Gen & Cheng, 1999; Kimms, 1999). Modeling multiple production lines as a multiple traveling salespersons problem (MTSP) and solving the problem with a GA has been largely unstudied. The production line modeling that has been done using multiple salespersons uses a matrix representation of the problem resulting in a dramatically increased solution space of size $(n + m – 1)!$ where m is the number of salespersons used (Lenstra & Rinnooy Kan, 1975; Tang, Liu, Rong & Yang, 2000). Another example of a MTSP is the vehicle scheduling problem that has been solved using GAs (Park, 2001; Malmborg, 1996; Baita, Pesenti, Ukovich & Favaretto, 2000; Ochi, Vianna, Drummond & Victor, 1998).

A specific example of a production problem being modeled as a multiple salesperson TSP can be found in the production departments of many newspaper companies. The post-press department of a newspaper's production department is responsible for inserting advertising into the printed paper and creating a product ready for delivery to various geographical regions called zones. As advertisers have different customer bases and target demographics, they can identify and pay for the zones where they wish their advertising to be delivered. Each of these zones has a different number of papers and therefore a different processing time on the post-press machinery. Also, due to the different inserts going to each area, a production stop of varying length is required to change from one zone to another. Each of these zones can be viewed as a city in a traveling salesperson problem.

Most newspapers run inserting equipment that utilizes multiple lines. Even the smallest newspaper production operation runs equipment that usually has two production lines. Larger papers may easily run four to six lines of production. Each of these production lines can be viewed as the individual salespersons in a multiple traveling salesperson problem. The goal for the production supervisor developing the schedule is to minimize the run time to process all of the newspapers. This is done by assigning the various zones (of various sizes) to the different production lines in such a way that balances the zones between the production lines and minimizes lost production time due to zone changes. Balancing work between the production lines and minimizing stops for zone changes completes the entire process in the minimum amount of time.

As part of this research an actual instance of the newspaper insert-scheduling problem is solved using a new genetic algorithm (GA) model. The model represents a four-line production operation as a two-part chromosome that a GA tool can solve simultaneously. To test the validity of the approach, an actual insert production problem from a mid-sized newspaper is used. The new modeling technique used produces good results when using commercially available GA packages. Details and results of this research are covered in Chapter 3 of this work.

ONE SALESPERSON WITH MULTIPLE TOURS

While the TSP is used as a model for many production scheduling problems, it has one feature which is not generally true for many production problems. A true TSP requires the solution to be a single continuous tour of all the cities, with the salesperson returning to his home city once all of the cities have been visited. In a production problem, returning to the home city usually is not significant, is added to the next tour, or can be done during non-production time.

The idea of using multiple salespersons to tour *n* cities is similar to the idea of using one salesperson to cover the cities, but with the tour broken up into many smaller tours with the salesperson returning home at the end of each of the small tours. The application of this can be easily seen in a company that has one salesperson to cover many cities where the salesperson leaves home on Monday, travels all week, and returns home on Friday. When considering the idea of using a single salesperson to complete multiple TSPs, the fact that each salesperson must return home must be included. The return to a home base is part of the tour being traveled and therefore affects the fitness of the solution. Also, if the trips to and from home are not included, the solution may violate constraints on the total distance (or time traveling) a salesperson is permitted on each trip.

This specific scheduling problem can be seen not only for "traveling salespersons" but also in other fields. For example, consider the scheduling issues faced by companies with multiple operating units (e.g., banks, hotels, restaurants) that use inspectors who must go out and evaluate some aspect of the company's operation. In the hotel industry, most companies have field inspectors who go to the individual properties to evaluate and rate various aspects of the property's condition and performance.

Chapter 4 of this research will examine the actual scheduling of hotel inspections for a national hotel chain. One inspector needs to minimize the distance traveled during his/her weekly tours that will ultimately result in all the properties in his/her region being inspected. With a problem of this nature, many constraints must be considered. In this research constraints include: time windows within which the properties must be inspected, inspector vacation and training days, maximum time traveled on each leg of the trip, new property training, and provisions to allow scheduling special case inspections. These issues add both soft and hard constraints to the MTSP problem.

FUTURE RESEARCH AND CONCLUSIONS

The idea of using multiple salespersons (or one salesperson on multiple tours) has many applications in various fields. Future directions for this research will explore other opportunities where this model will prove useful. The research into solving this model only scratches the surface as to different methodologies that may be used. The potential for new areas of application and model solving are covered in Chapter 5.

CHAPTER 2

A New Approach to Solving the Multiple Traveling Salesperson Problem Using Genetic Algorithms

INTRODUCTION

The multiple traveling salesperson problem (MTSP) is a difficult problem that can be used to model a significant number of practical problems. The MTSP is similar to the notoriously difficult traveling salesperson problem (TSP) that seeks an optimal tour of *n* cities, visiting each city exactly once with no sub-tours. In the MTSP, the *n* cities are partitioned into *m* tours, with each tour assigned to a separate salesperson. The MTSP is even more difficult than the TSP because it requires determining which cites to assign to each salesperson, as well as the optimal ordering of the cities within each salesperson's tour.

Perhaps the most common application of the MTSP is in the area of scheduling. The scheduling of jobs on a production line is often modeled as a TSP. If the production operation is expanded to have multiple parallel lines to which the jobs can be assigned, the problem can be modeled as a MTSP (Carter & Ragsdale, 2003). Another scheduling problem that is often modeled as an MTSP is the vehicle-scheduling problem (VSP). The VSP consists of scheduling a set of vehicles, all leaving a common depot, to visit a number of locations such that each location is visited exactly once (Park, 2001).

A variation on the TSP that can also be modeled as a MTSP involves using one salesperson to visit all *n* cities in a series of *m* smaller tours. This describes the scheduling problem of sales/service personnel that visit *n* cities over a period of time but travel during the week and return home on weekends.

Due to the combinatorial complexity of the MTSP, it is necessary to employ heuristics to solve problems of realistic size. Genetic algorithms (GAs) represent a class of heuristic search techniques for the TSP that has been researched for a number of years. More recently, researchers studying the VSP have expanded the use of GAs developed for the TSP to also solve the MTSP.

Researchers working on solving the VSP using GAs have focused on using two different chromosomes for the MTSP. Both of these chromosomes can be manipulated using classic GA operators developed for the TSP. This research introduces a new chromosome for the MTSP that works with classic GA TSP operators but dramatically reduces the number of redundant solutions in the solution space, thereby improving the efficiency of the search.

The remainder of this research will be organized as follows. First we review the literature on solving the TSP and MTSP using GAs. Next, we provide an overview of GAs focused on the characteristics associated with well-designed chromosomes. Next, the two chromosomes traditionally used for the MTSP are presented and contrasted with our new proposed chromosome. A computational comparison of the various chromosomes is then presented followed by concluding remarks and suggested areas for future research.

LITERATURE REVIEW

A number of different methods have been proposed for obtaining either optimal or near optimal solutions for the TSP. The methods used to solve the TSP range from classic methodologies based on linear programming (Wong, 1980) and branch-and-bound (Little, Murty, Sweeney and Karel, 1963; Bellmore & Malone, 1971) to artificial intelligence methods such as neural networks (Shirrish, Nigel & Kabuka, 1993), tabu search (Glover, 1990), and GAs (Goldberg & Lingle, 1985). For a good overview of the TSP and various proposed solutions methodologies for this problem see Lawler, Lenstra, Rinnooy Kan, & Shimoys (1985).

Given the combinatorial complexity of TSPs of realistic size, a solution methodology that efficiently solves TSPs to global optimality remains elusive. While advances have been made in solving the TSP, those advances have come at the cost of more complicated computer code (Chatterjee, Carrera & Lynch, 1996).

The work on solving MTSPs using GAs has focused on vehicle scheduling problems (Malmborg, 1996; Park, 2001). The vehicle scheduling problem (VSP) consists of scheduling a fleet of *m* vehicles to visit *n* cities with each city being visited by one and only one vehicle. The VSP typically includes constraints on the number of cities each vehicle can visit due to the capacity of each truck available and the size load to be picked up at each city. In some cases, the cities must be visited between specific times called time windows. These issues lead to a number of different possible configurations for the VSP: VSP with/without time windows, VSP with heterogeneous/homogeneous vehicle capacities, and VSP for minimum distance/minimum vehicle requirements. A variety of objectives can also be considered, including: minimize the total distance, minimize the number of vehicles required, and minimize lateness (if time windows are used).

OVERVIEW OF GAs

Genetic algorithms (GAs) are a relatively new optimization technique that can be applied to TSPs. The basic ideas behind GAs evolved in the mind of John Holland at the University of Michigan in the early 1970s (Holland, 1975). GAs were not originally designed for highly constrained optimization problems but were soon adapted to order-based problems like TSP (Goldberg & Lingle, 1985; Jog, Suh and Van Gucht, 1989; Whitley, Starkweather & Fuquay, 1989). The development of effective GA operators for TSPs led to a great deal of interest and research to improve GAs' performance for this type of problem (Poon & Carter, 1995; Qu & Sun, 1999; Katayma & Sadamoto, 2000). Several reviews of solving TSPs with GAs have been published that provide comprehensive reviews of the operators and associated issues (Potvin, 1996; Schmitt & Amini 1998; Schaffer, Eshelman, & Offutt, 1991.).

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are then evaluated according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem (Ragsdale, 2001). The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes, or until no significant improvement occurs in the population for a given number of iterations.

The key to finding a good solution using a GA lies in developing a good chromosome representation of solutions to the problem. A good GA chromosome should reduce or eliminate redundant chromosomes from the population. Redundancy in the chromosome representation refers to a solution being able to be represented in more than one way and appearing in the population multiple times. These multiple representations increase the search space and slow the search.

A well-designed chromosome must accurately represent a solution to the problem and allow the GA operators to work effectively to generate better solutions as the iterative evolutionary process continues. An example of a chromosome representation designed for the efficient solution of a specific problem is found in the grouping genetic algorithm developed by Falkenauer (1998). Falkenauer's grouping genetic algorithm increases the effectiveness of GAs on grouping problems (bin packing, workshop layout and graph coloring) by modeling the problem in a way that reduces the redundancy in the population. A well-designed chromosome must also allow its fitness to be calculated easily. This facilitates comparison of the various chromosomes to determine which are better and should remain in the population.

Solving the TSP using GAs has generated a great deal of research focused on how best to perform the action of "evolving" an optimal (or good) solution to the problem. The operators used in the GA are key to finding a good solution to the problem. The TSP is unique in that it requires the solution to have each city visited once and only once with no sub-tours. This requires the operators to perform crossover and generate the new child solutions while maintaining feasibility.

A number of different crossover methods have been proposed in the literature to solve the TSP using a GA. Some of the most commonly used operators include: Order Crossover (Goldberg, 1989; Davis, 1985; Starkweather, Whitley, Whitley, & Mathial, 1991; Oliver, Smith & Holland, 1987), Partially Mapped Crossover (Goldberg, 1989; Starkweather, et al, 1991; Goldberg and Lingle, 1985; Oliver, Smith & Holland, 1987), Cycle Crossover (Goldberg, 1989; Starkweather, et al, 1991; Oliver, Smith & Holland, 1987) and Asexual Crossover (Chatterjee, et al, 1996; Schmitt and Amini, 1998; Fox & McMahon, 1991). Other TSP operators that have been proposed include utilizing a matrix chromosome representation (Poon and Carter, 1995; Qu and Sun, 1999), hybrid operators (Laiw, 2000), simple crossover (Knosala & Wal, 2001), ordered crossover #2 (Syswerda, 1991), moon crossover (Moon, Kim, Choi & Seo, 2002), and position based crossover (Syswerda, 1991).

CHROMOSOME REPRESENTATIONS FOR THE MTSP

As mentioned earlier, two different chromosomes are commonly used when solving the MTSP using GAs. We discuss these chromosomes below, discuss their properties and weaknesses, and then introduce a new chromosome that offers several advantages for solving the MTSP.

The Two Chromosome Technique

Figure 2-1 illustrates one approach for representing solutions to a MTSP (where n=15 and m=4) that we refer to as the "Two Chromosome" technique. As the name implies, this method requires two chromosomes, each of length n, to represent a solution. The first chromosome provides a permutation of the n cities while the second chromosome assigns a salesperson to each of the cities in the corresponding position of the first chromosome (Malmborg, 1996; Park, 2001).

Cities

2	5	14	6	1	11	8	13	4	10	3	12	15	9	7

Salespersons

2	1	1	3	4	3	2	4	4	1	3	2	1	2	3

Figure 2-1
Example of the Two Chromosome Representation
for a 15 City MTSP with 4 Salespersons.

In the example in Figure 2-1, cities 2, 8, 12 and 9 (in that order) are visited by salesperson 2. Cities 5, 14, 10 and 15 (in that order) are visited by salesperson 1; and likewise for the remaining cities being assigned to salespersons 3 and 4. Using this chromosome design, there are $n!m^n$ possible solutions to the problem where n is the number of cities and m is the number of salespersons. However, many of the possible solutions are redundant. For example, the first two genes in each of the above chromosomes can be interchanged to create a different chromosome that results in an identical (or redundant) solution.

The One Chromosome Technique

Figure 2-2 illustrates a second method for representing solutions to the same MTSP (where *n*=15 and *m*=4). This technique involves using a single chromosome of length *n*+*m*–1 and is referred to as the "One Chromosome" technique (Tang, Liu, Rong & Yang, 2000). In this technique, the *n* cities are represented by a permutation of the integers from 1 to *n*. This permutation is partitioned into *m* sub-tours by the insertion of *m*-1 negative integers (from –1 to –(*m*–1)) that represent the change from one salesperson to the next. Any permutation of these *n*+*m*–1 integers represents a possible solution to the problem.

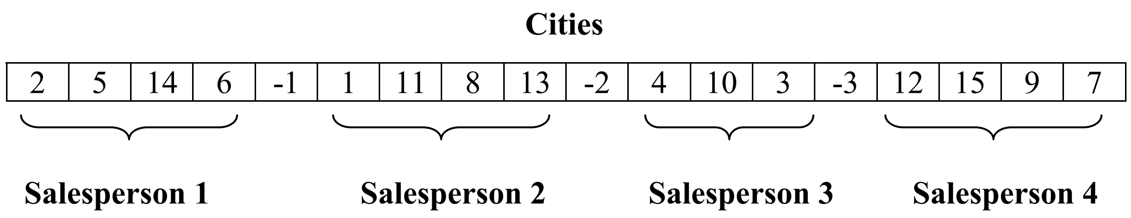


Figure 2-2
Example of the One Chromosome Representation
for a 15 City MTSP with 4 Salespersons.

In the example in Figure 2-2, the first salesperson would visit cities 2, 5, 14 and 6 (in that order). The second salesperson would visit cities 1, 11, 8 and 13 (in that order), and so on for salespersons 3 and 4. Using this chromosome scheme, there are (*n*+*m*–1)! possible solutions to this problem. Many of the possible chromosomes are redundant. For example, simply exchanging the values of the first five genes with those of the next five genes would produce an equivalent (redundant) solution. Additionally, under this approach it is possible for two or more of the negative integers to appear consecutively – effectively reducing the number of salespersons to be utilized.

The Two-Part Chromosome Technique

Figure 2-3 illustrates a new chromosome for the same MTSP (with *n*=15 and *m*=4) that has two distinct parts; hence the name "Two-Part Chromosome." The idea of using a two-part chromosome for the MTSP is similar to the two-part chromosome of the grouping genetic algorithm (Falkenauer, 1998). The first part of the chromosome is a permutation of integers from 1 to *n*, representing the *n* cities. The second part of the chromosome is of length *m* and represents the number of cities assigned to each of the *m* salespersons. The values assigned to the second part of the chromosome are constrained to be *m* positive integers that must sum to the number of cities to be visited (*n*).

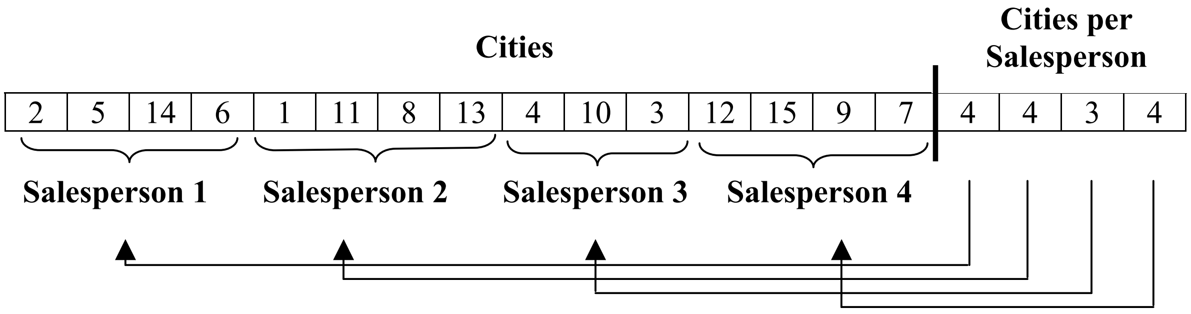


Figure 2-3
Example of the Two-Part Chromosome Representation for a 15 City MTSP with 4 Salespersons.

In Figure 2-3, salesperson 1 visits cities 2, 5, 14 and 6 (in that order), salesperson 2 visits cities 1, 11, 8 and 13 (in that order), and so on for salespersons 3 and 4. Using the new two-part chromosome method reduces the size of the search space due to the elimination of some (but not all) redundant solutions. For example, cities assigned to salesperson 1 always appear first in the two-part chromosome followed by the cities assigned to the second salesperson. This was not the case in either of the previous two chromosomes, where cities for a given salesperson could appear in any relative position in the chromosome (accounting for much of the redundancy in those chromosomes.)

Using the two-part chromosome for the MTSP, there are $n!$ possible permutations for the first part of the chromosome. The second part of the chromosome represents a positive vector $(x_1, x_2, \ldots, x_m)$ that must sum to n. There are $\binom{n-1}{m-1}$ distinct positive integer-valued m-vectors that satisfy this requirement (Ross, 1984). Therefore, the solution space for the two-part chromosome is of size $n!\binom{n-1}{m-1}$.

COMPARISON OF SOLUTION SPACES

The three equations that describe the size of the solution space for each of the above chromosomes are summarized below. In the appendix, we present mathematical proofs showing that the solution space for the two-part chromosome is smaller than those of the other two techniques when $n>m>1$.

Technique	**Solution Space**
Two Chromosome:	$n!m^n$
One Chromosome:	$(n + m - 1)!$
Two-Part Chromosome:	$n!\binom{n-1}{m-1}$

Figure 2-4 illustrates the magnitude of the differences in the solution spaces for the three chromosomes for a MTSP with n=15 cities as the number of salespersons is varied from 1 to 14. Note that the y-axis on this chart represents the natural logarithm of the size of the solution space. When there is one salesperson (m=1), the chromosomes default to a single TSP with identical solution spaces (*i.e.*, n!=15!=1.30767E+12). As the number of salespersons (m) increases, the solution spaces of the traditional chromosomes quickly dwarf that of the new two-part chromosome. Of course, there is some extra computational overhead associated with ensuring that the second portion of the two-part chromosome satisfies the constraint imposed on it. However, our computational results (presented below) suggest this overhead is worth the investment, particularly for large (more difficult) MTSPs.

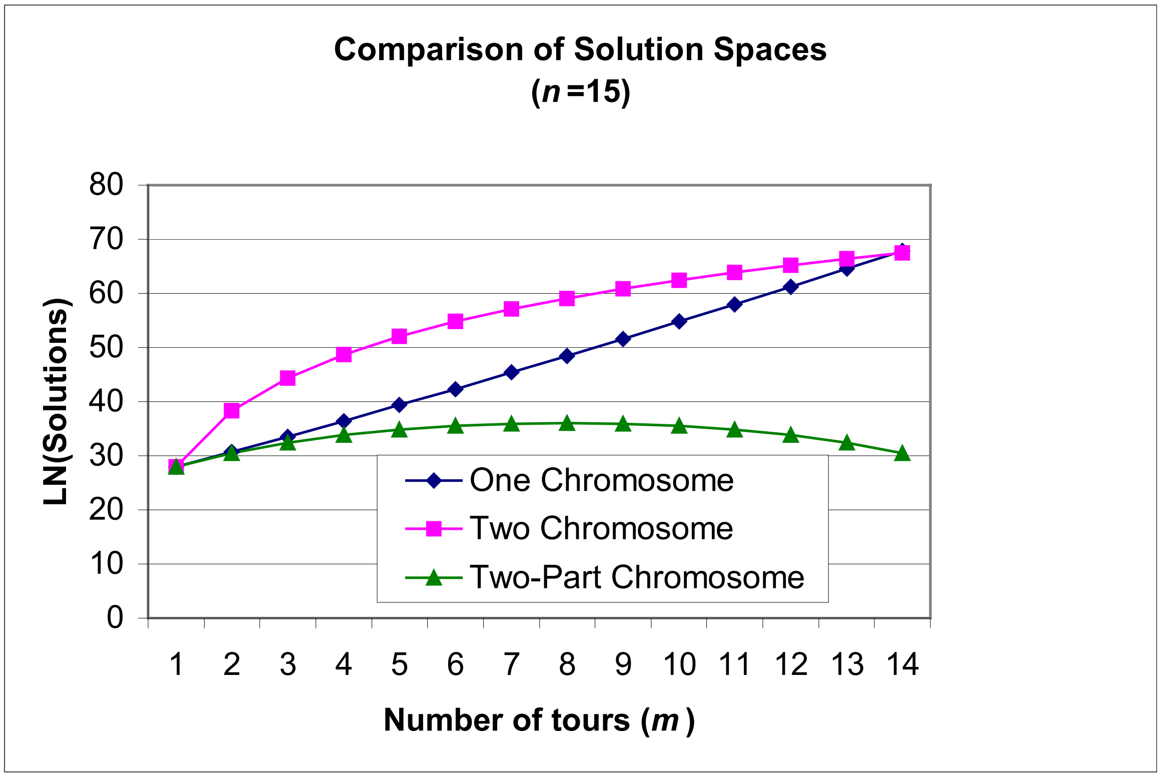


Figure 2-4
Comparison of Chromosome Solution Spaces
for a 15 City MTSP with *m* Salespersons.

COMPUTATIONAL TESTING METHODOLOGY

In order to evaluate the practical benefits of the proposed two-part chromosome, computational tests were conducted to compare the performance of all three techniques on a set of problems created for the MTSP. The test problems were selected from a standard collection of TSPs with 51, 100, and 150 cities (Reinelt, 2001) that were transformed into MTSPs by using more than one salesperson (*m*) to complete the tour. Table 2-1 summarizes the experimental conditions of the twelve different problem size (*n*), salesperson (*m*) combinations along with the run times allowed for each type of problem. For each level of *n*, all salespersons started and ended their individual tours in the same city.

Number of Cites (*n*)	**Number of Salespersons (*m*)**	**Run Time (in minutes)**
51	3, 5, and 10	5
100	3, 5, 10, and 20	10
150	3, 5, 10, 20, and 30	15

Table 2-1
Computational Test Conditions

Two different fitness (or objective) functions were considered. The first fitness function minimized the total distance traveled by all of the salespersons combined. For these runs, each salesperson was required to visit at least one city (other than the home city). This objective would represent a situation where there is a set number of salespersons to be used and there are no constraints associated with the maximum number of cities visited by any one salesperson. The second fitness function minimized the longest route of the *m* salespersons. This objective function would be used when scheduling to equalize the workload between available salespersons.

Each of the twelve problem size (*n*), salesperson (*m*) combinations were run 10 times for each of the two different fitness objectives. The tests were run on a Pentium 4, 1.6Gz desktop machine with 256 MB of RAM using programs written in Visual Basic.

GA Issues

The GA programs used for testing all utilized the ordered crossover method, roulette wheel selection, a population of 100 and a 5% mutation probability. Ordered crossover was selected as it is a commonly used TSP operator. The values for population and mutation were select to represent commonly used values. A detailed explanation of the operation of roulette wheel selection ordered crossover, population size and mutation operators can be found in Goldberg (1989).

The second part of the two-part chromosome used a single point asexual crossover method. This method simply cut the second part into two sections and reversed the order in which the two pieces were arranged. This type of crossover was necessary to ensure that the second part remained feasible (the sum of the values in the chromosome equaling *n*).

The creation of the initial populations for each of the three solution techniques had to be handled differently due to the varying structure of the chromosomes. The initial population generation strategies are summarized below.

<u>Two Chromosome Technique</u> - The first chromosome is a randomly generated permutation of the *n* cities. The second chromosome is created by filling each position with a randomly generated number between one and the number of salespersons (*m*).

<u>One Chromosome Technique</u> - The initial chromosomes are generated by randomly generating a permutation of the *n* cities plus *m*-1 negative integers to represent the change from one salesperson to another.

<u>Two-Part Chromosome Technique</u> - The first part of the chromosome is a randomly generated permutation of the *n* cities. Recall that the sum of the *m* positive integers in the second part of the chromosome must be *n*. For the tests using the fitness function representing the total distance of all salespersons, the value for gene *i* (x_i) in the second part of the chromosome was generated as a discrete uniform random number between 1 and $n - \sum_{k=1}^{i-1} x_k$, for *i* = 1 to *m* (*i.e.*, the maximum value of each successive gene was based on *n* and the sum of the values already appearing in the second part of the chromosome). For the tests using the fitness function that minimized the longest of the *m* routes, *n* discrete uniform random numbers from 1 to *m* were generated and counted in their corresponding positions in the second part of the chromosome (i.e., $x_i = x_i + 1$ when the value *i* was generated). Also, when the GA created new child chromosomes for this technique, the first part of the child chromosome was matched with each of the second parts from the existing population and with the newly created second part and the fitness determined for each pairing. The new child solution was then matched with the second part that resulted in the best fitness value for the new two-part chromosome.

Computational Results: Minimum Total Distance

When minimizing the total distance of all the salespersons in a MTSP, as the number of salespersons increases, the total distance of all of the trips also tends to increase. This results from the fact that each salesperson must start and return to the home city. So as the number of salespersons increases, the number of trips into and out of the home city increases, thereby increasing in the total distance traveled. The results for the runs using the total distance fitness function are presented in Figures 2-5 though 2-10.

Salespersons	One Chromosome (miles)	Two Chromosome (miles)	Two-Part Chromosome (miles)
3	596[a]	951[a]	651
5	726	1180[a]	691
10	1015[a]	1403[a]	843

Figure 2-5
Averages for 51 City Problem Minimizing Total Distance.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.02 level.

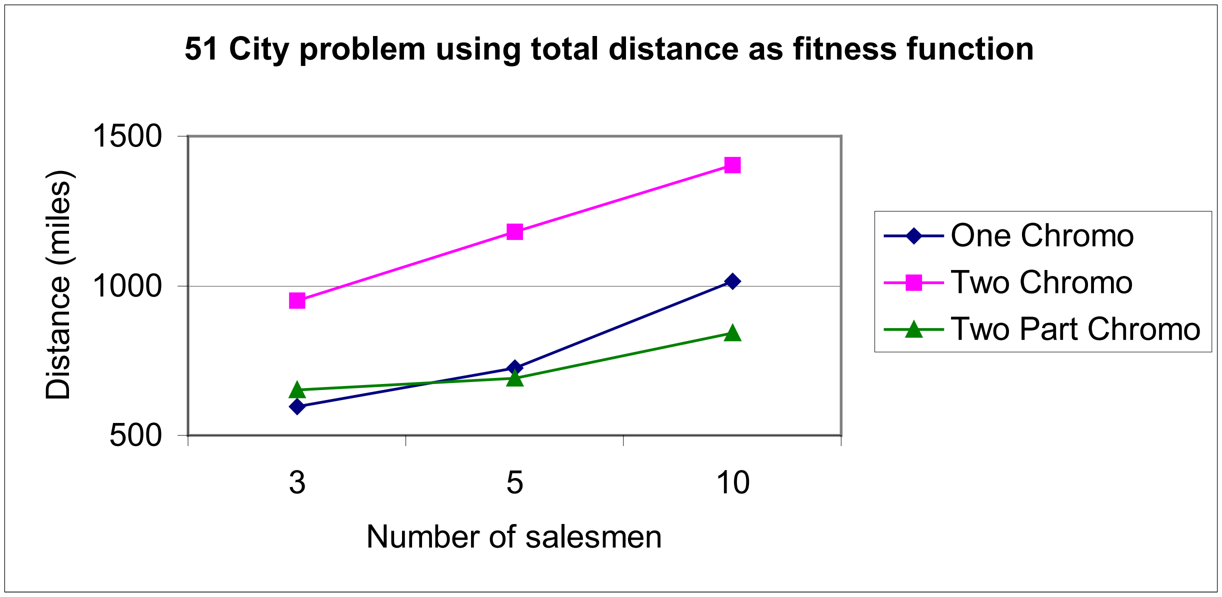


Figure 2-6
51 City Problem Using Total Distance as Fitness Function

Salespersons	One Chromosome (miles)	Two Chromosome (miles)	Two-Part Chromosome (miles)
3	46,929[a]	80,195[a]	65,023
5	51,509[a]	97,741[a]	65,525
10	68,664[a]	129,000[a]	76,537
20	115,210[a]	151,935[a]	91,196

Figure 2-7
Averages for 100 City Problem Minimizing Total Distance.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.02 level.

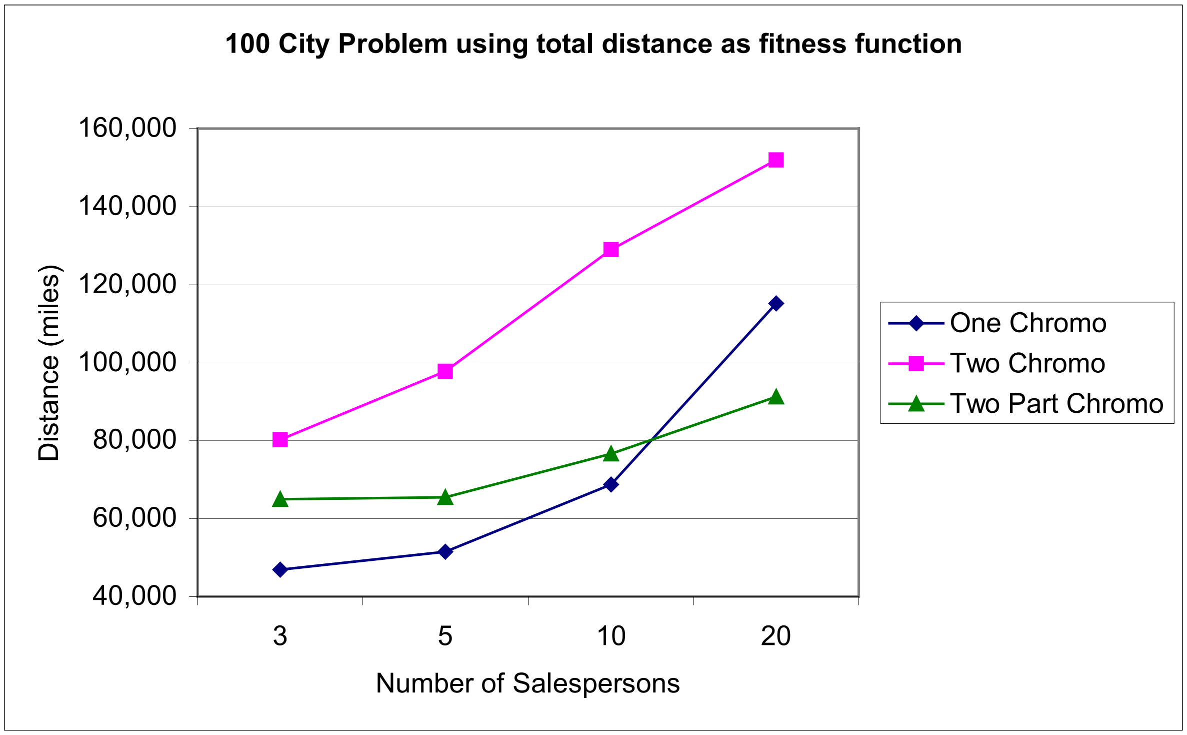


Figure 2-8
100 City Problem Using Total Distance as Fitness Function

Salespersons	**One Chromosome (miles)**	**Two Chromosome (miles)**	**Two-Part Chromosome (miles)**
3	17,754[a]	37,373[a]	24,853
5	20,558[a]	38,766[a]	24,930
10	27,621	39,906[a]	25,646
20	43,279[a]	41,484[a]	28,799
30	56,006[a]	45,364[a]	29,651

Figure 2-9
Averages for 150 City Problem Minimizing Total Distance.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.02 level.

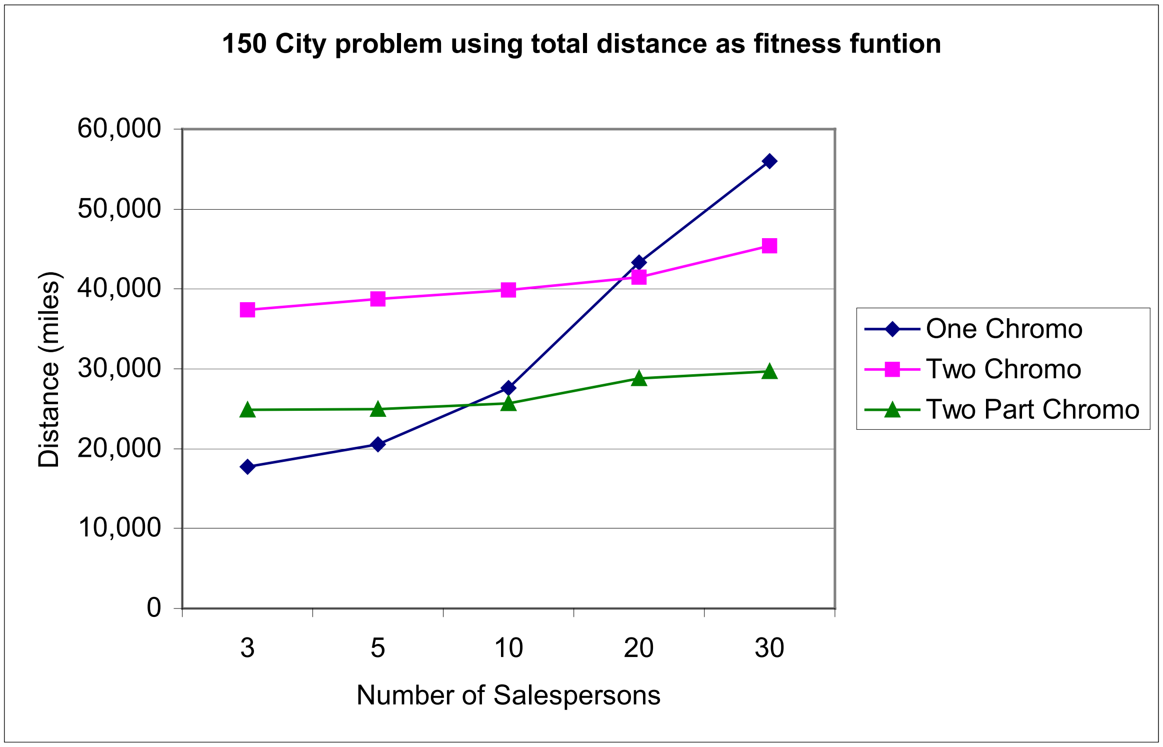


Figure 2-10
150 City Problem Using Total Distance as Fitness Function

The results from the tests using total distance as the fitness function do not show a superior technique for all cases. The trend for each of the different problems was that the one chromosome method produced the best results with a small number of salespersons, but as the number of salespersons increased (and the solution space grew), the two-part method began to exhibit superior performance. In every case, the two-part method produced statistically better results than the two chromosome method. With the largest problem (*n*=150 cities), the trend for the two-part method to surpass the one chromosome method was even more dramatic than for the smaller problems. For the 150 city problem, even the two chromosome method surpassed the one chromosome method in the 20 and 30 salesperson configurations. These results strongly suggest that as problem complexity increases, the proposed two-part chromosome is the preferred solution alternative.

Computational Results: Minimum Longest Tour

Next, we repeated our tests using the objective of minimizing the longest individual salesperson's tour. With this objective, the fitness values decrease as the number of salespersons increase because we are dividing the cities up among more salespersons. This creates a situation where each salesperson visits fewer cities, so the fitness values decreases (improve) with the addition of more salespersons. The results of these tests are presented in Figures 2-11 through 2-16.

Salespersons	One Chromo (miles)	Two Chromo (miles)	Two-part Chromo (miles)
3	329[a]	384[a]	275
5	277[a]	276[a]	202
10	208[a]	183[a]	145

Figure 2-11
Averages for 51 City Problem Minimizing the Longest Tour.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.001 level.

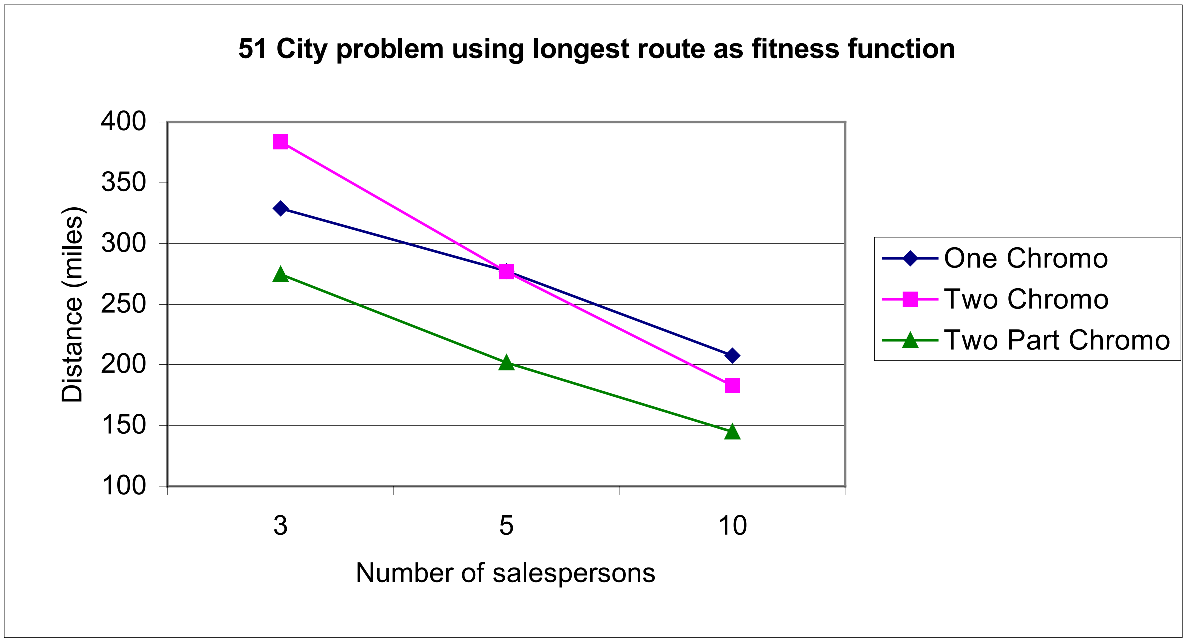


Figure 2-12
51 City Problem Using Longest Route as Fitness Function

Salespersons	One Chromosome (miles)	Two Chromosome (miles)	Two-Part Chromosome (miles)
3	30,733[a]	35,203[a]	26,499
5	26,005[a]	26,764[a]	19,200
10	17,522[a]	16,525[a]	13,022
20	12,971[a]	10,902[a]	9,413

Figure 2-13
Averages for 100 city Problem Minimizing the Longest Tour.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.005 level.

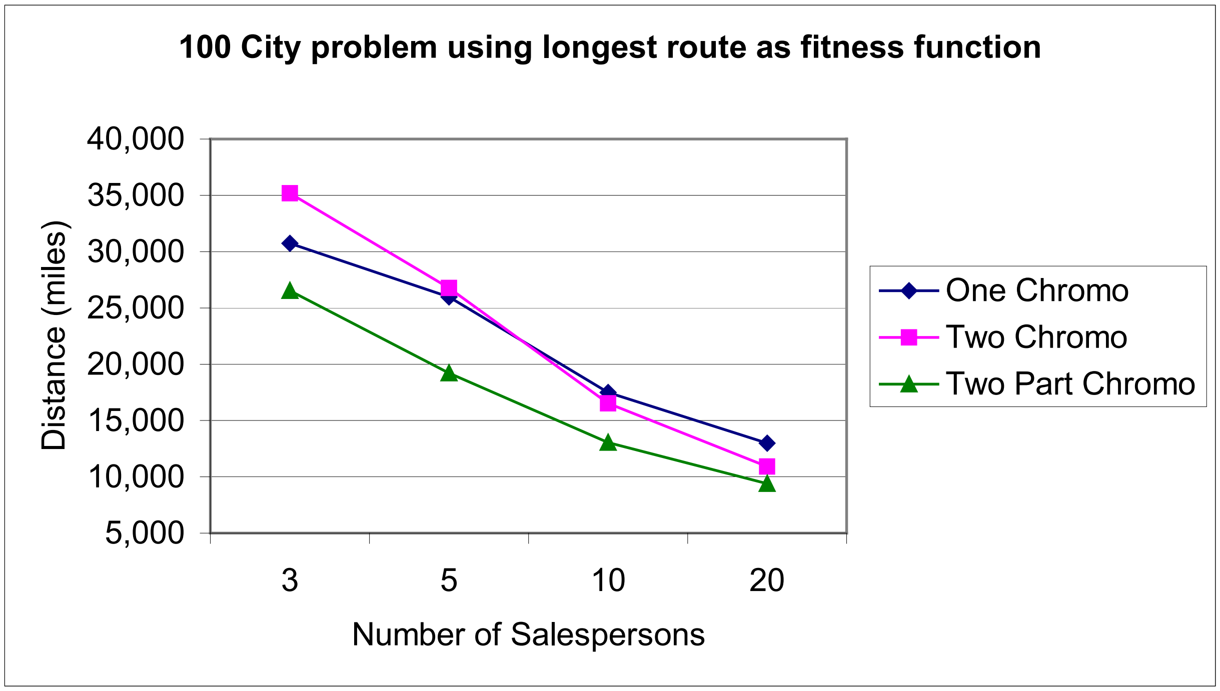


Figure 2-14
100 City Problem Using Longest Route as Fitness Function

Salespersons	One Chromosome (miles)	Two Chromosome (miles)	Two-Part Chromosome (miles)
3	11,455[a]	13,882[a]	10,055
5	8,916[a]	9,456[a]	7,128
10	5,820[a]	5,506[a]	4,598
20	4,284[a]	3,473[a]	2,777
30	3,795[a]	3,482[a]	2,504

Figure 2-15
Averages for 150 City Problem Minimizing the Longest Tour.

a: Indicates statistically significant differences from the Two-Part Chromosome at the 0.001 level.

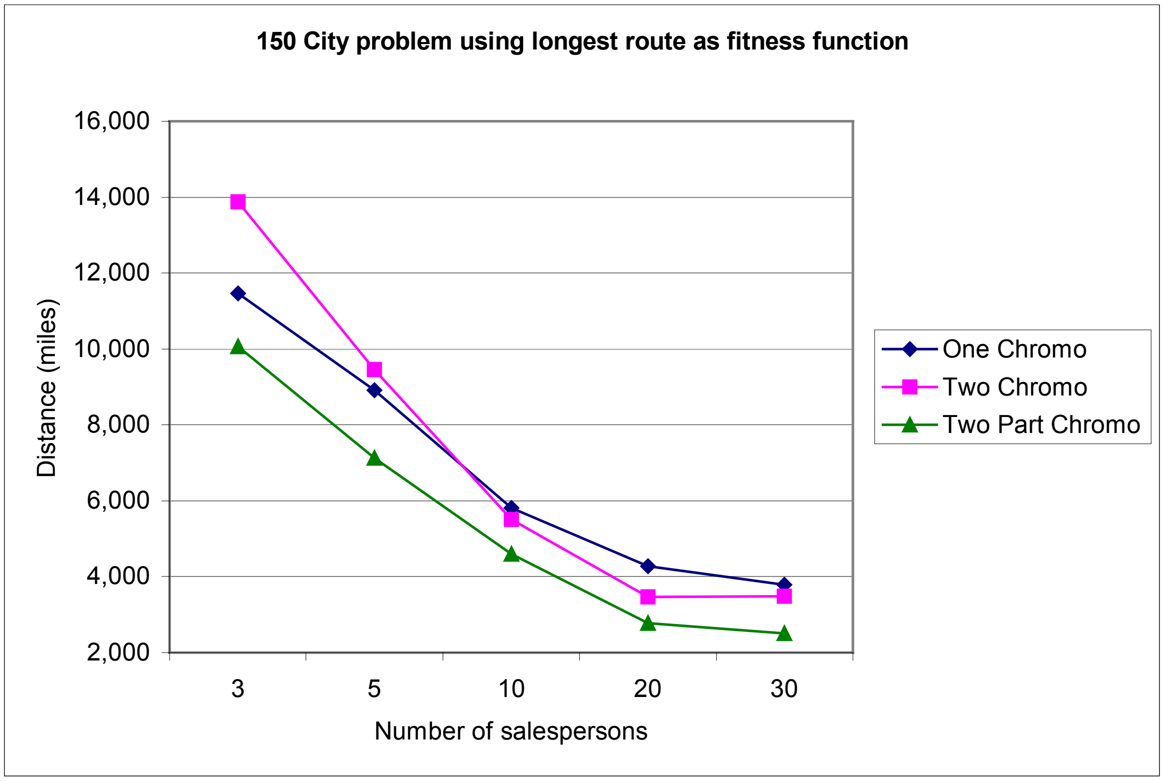


Figure 2-16

150 City Problem Using Longest Route as Fitness Function

The results from the testing conducted pursuing the objective of minimizing the longest tour clearly show the two-part technique to be the superior method. In every problem, for all levels of salespersons used, the two-part chromosome method found a statically lower fitness value than either of the other two methods. The smaller solution space associated with the two-part chromosome clearly provides an advantage when searching for a balanced distribution of the workload among the salespersons. This can be of particular interest in areas such as production scheduling (balancing the workload amongst the available production lines) and in the VSP (balancing the load amongst the available trucks).

CONCLUSIONS

Modeling the MTSP using the new two-part chromosome proposed in this paper has clear advantages over using either of the existing one chromosome or the two chromosome methods. The two-part chromosome's smaller solution space allows it to produce solutions with better fitness values in most cases tested in this research (and all of the most difficult cases). When minimizing the total distance traveled as the fitness function, the two-part chromosome was superior when the number of salespersons grew sufficiently large. When pursuing the "balance workload" objective of minimizing the longest tour, the two-part chromosome produced superior results as compared to either of the other methods in every case.

The two-part chromosome proposed in this paper has a number of areas of usage that will be of interest for future research. The two-part chromosome has performed well in theoretical and empirical comparisons to existing chromosomes for the MTSP. Additional research needs to be conducted in the area of VSP, taking into account the various constraints and objectives the VSP faces. Also, a variety of production scheduling problems may benefit from the application of the new two-part chromosome introduced in this research.

Chapter 3

A Production Scheduling Problem Modeled as a Multiple Salesperson TSP

INTRODUCTION

The TSP can be used to model many different production scheduling problems. By relaxing the constraint that only one salesperson is used to visit all of the cities there are additional problems that can be modeled, or existing problems that can be modeled more accurately. The newspaper industry faces a TSP type problem when developing schedules for the inserting production run of the newspaper. This is a problem due to the time constraints on producing the newspaper and the huge amount of money the newspapers receive from the advertising.

Advertising is the primary source of revenue for newspaper companies. The Scripps Company owns 19 daily newspapers producing over $900 million in annual revenue. For 1999, Scripps (E. W. Scripps, 1999) reported that advertising was 76.1% of total newspaper revenues. Over the last 10 to 15 years the newspaper industry has been adjusting to changes in the mix of services that produce this revenue. The two main categories of services are run on press (ROP) advertising and pre-printed insert advertising. ROP advertising is printed in the newspaper during the live press run each night while pre-printed inserts are produced (usually at a commercial printing facility) before the nightly production run and inserted into or delivered with the newspaper.

While revenue has been increasing in both categories of advertising, revenue from pre-printed inserts has been growing at a higher rate than ROP advertising (E. W. Scripps, 1999). For many newspaper companies, this shift in revenue mix has created scheduling challenges in the production area. With inserts, advertisers can select the zones to which specific sets of advertisements are distributed. A **zone** is a distinct geographical area where all the papers delivered in the area receive the same set of inserts. The challenge for newspaper companies is to schedule the production run to process the correct combination of inserts for all the different zones and complete the run early enough to get the papers to the circulation department (and readers) on time. For many papers, the problem is exacerbated by advertisers' desire to "micro-zone" or have more zones of smaller size, increasing the specificity with which different groups of consumers can be targeted.

The task of developing a production schedule for this problem can be characterized as a traveling salesperson problem (TSP) with multiple salespersons (production lines) being available to visit the required cities (or in our case zones). The problem here is one of optimizing the salesperson's routes while balancing the routes as equally as possible between the salespersons. Newspapers typically have little expertise with optimization techniques for solving such difficult problems and often rely on manual scheduling heuristics.

This research explores the challenges associated with insert scheduling and develops an optimization technique utilizing a genetic algorithm (GA). The remainder of this paper is organized as follows. First, we provide a detailed look at the reasons for the increase in pre-printed advertising inserts and the effects this has had on newspaper companies. Next, we provide an overview of GAs and discuss how they can be used to solve this type of problem. We then describe the insert scheduling problem for an actual newspaper company and show how this problem can be modeled using a spreadsheet. Finally, we use two commercial GA add-ins to optimize the spreadsheet model and evaluate their performance against the actual solution created and implemented by the newspaper company.

ADVERTISING INSERTS AND THE NEWSPAPER INDUSTRY

Information technology has made it possible for advertisers to have a great deal more information about their current and potential customers. Massive customer databases combined with data mining techniques have allowed advertisers to identify the demographics of their customer base and identify the geographic areas where people with similar characteristics tend to live (NAA, 2001). Armed with this information, advertisers are trying to get the most "bang for the buck" out of each advertising dollar spent. As a result, many large advertisers have reduced or entirely removed ROP advertisements from the newspaper. These advertisers are instead opting for pre-printed inserts. In 1997, the newspaper industry saw total revenue for inserts surpass the revenue from ROP ads for the first time since such data has been tracked (Neuwirth, 1998).

Pre-printed inserts offer several advantages for advertisers. Different sizes and quality of paper stock can be used to make ads unique and more colorful than possible on a newspaper printing press. Also, advertisers can tightly control quality for pre-printed inserts unlike newspaper quality that varies widely (Neuwirth, 1998). These pre-printed pieces can also be used to do mailings to areas where the paper is not delivered. With inserts, advertisers can specify the area(s) where the inserts will be distributed within the framework of the newspaper's zoning capability.

Newspaper companies commonly adopt each postal zip code to be a unique zone. Because advertisers can select the zones they desire to target, a unique set of advertising inserts can be delivered with newspapers to each zone. Insert zoning has allowed advertisers to increase the effectiveness and reduce the cost of their total advertising effort by hitting areas with high response potential with a colorful, high quality advertisement and skipping less promising areas. Not surprisingly, advertisers are continually pushing newspaper companies to create micro-zones to allow them to reduce their advertising costs while enhancing the desired effect. The loads associated with each zone are assembled with inserting equipment in the post-press production operation. A typical production layout utilizing four inserting lines is shown in Figure 3-1.

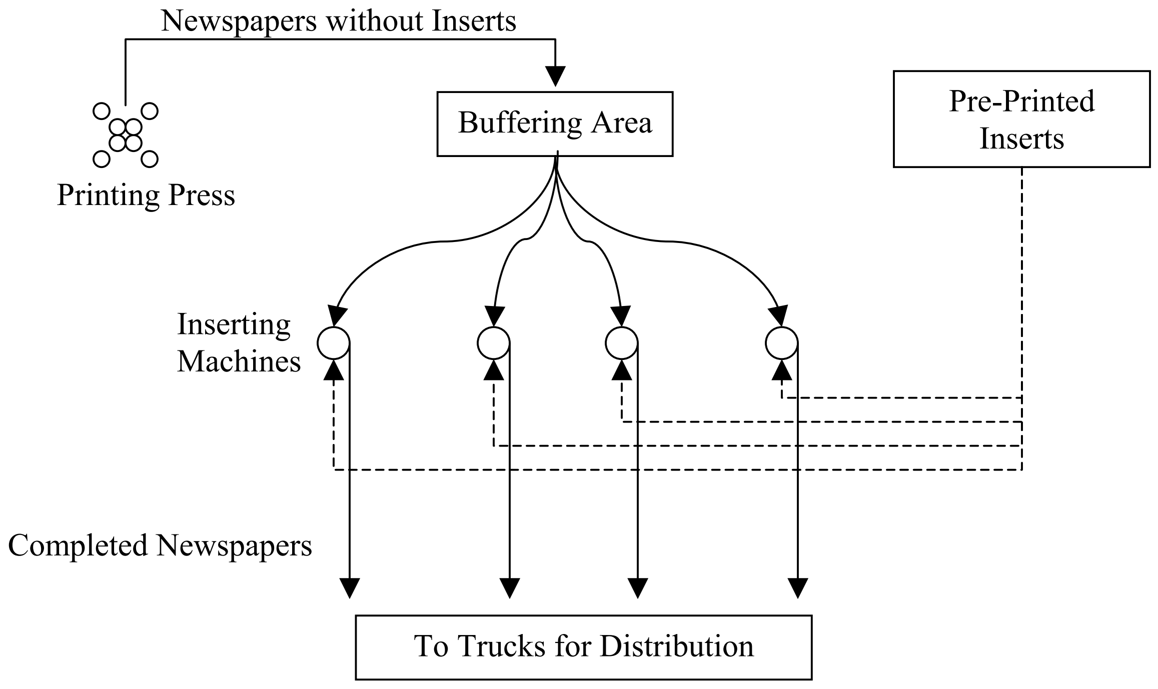


Figure 3-1
Newspaper Production Flow Diagram

Pre-printed inserts and the push toward micro-zoning have created new challenges for the post-press production operations at many newspaper companies. As zones have gotten smaller, the complexity of the nightly production runs has increased greatly because some types of inserting equipment must be stopped to changeover from one zone's set of inserts to the next. The push for more and smaller zones could drive an operation with 50 zones to possibly having several hundred micro-zones, greatly increasingly the complexity of production scheduling.

While micro-zoning increases the complexity of assembling a newspaper, other pressures are simultaneously pushing the production department to reduce the overall cycle time. For instance, the news department wants the latest possible deadline to allow the newspaper to include late breaking news, late sports scores, and lottery results. All of these things are important to various reader segments and, therefore, important to keeping readers. At the same time, the circulation department wants early production finish times to allow them to deliver the paper as soon as possible for those readers who want to read the newspaper at breakfast or on their way to work. So there is a squeeze from both sides on the production department to shorten its cycle time while the number of zones (and workload) is actually increasing.

Historically, newspaper production departments have responded in one of two ways to the problems created by micro-zoning and the associated pressures from other departments. One response is to replace existing post-press equipment with new large capacity machinery that can automate insert zone changes. This option eliminates the changeover time between zones, but at a significant cost. A state of the art inserting machine supporting automatic zone changes typically costs on the order of $1.5 to $2.0 million each with the associated auxiliary equipment, plus possible building construction costs. Thus, new machinery is not a viable solution to the insert scheduling problem for all newspaper companies.

Despite the improved micro-zoning capabilities afforded by new inserting machines, newspaper companies in the U.S. are limited in their ability to pass the cost of this equipment on to advertisers due to competitive pressure from the United States Postal Service (Editor and Publisher, 1998). Insert pricing must be kept at a level that beats the cost of bulk mailing the advertising pieces. If the price for both methods of delivery becomes comparable, advertisers often prefer bulk mailing. Bulk mailing has the advantage that the advertising piece goes to every household as opposed to only the subscribers of the newspaper. Also, bulk mailing can be broken up into geographical areas not permitted by the zones newspaper companies use.

Another way that newspaper production departments have responded to the cycle time squeeze they face is to assemble packages of advertising inserts for each zone during the dayshift and combine them with the newspaper during the delivery process. By removing the inserting operation from the nightly production run, this option allows newspaper companies to accommodate smaller zones while not purchasing new (expensive) equipment. However, this also creates the need for additional production workers during the day and doubles the number of products that must be handled by the circulation department. So while this option keeps capital costs down, it tends to increase operational expenses (namely labor). And even if advertising inserts are assembled during the day, if hundreds of micro-zones are involved, there is still a need to schedule the insert assembly process as efficiently as possible to minimize labor requirements and ensure the product is available when needed.

To avoid the costs associated with purchasing new inserting equipment or scheduling additional workers during the day, newspaper companies must schedule the nightly production run in the most efficient manner. On "light" nights where most zones receive the same set of inserts, it may be easy to schedule a run that can be completed within the required time. However, on "heavy" nights where most zones receive different sets of inserts, it can be extremely difficult to devise an efficient production run -- especially if multiple zones and production lines are involved.

The insert scheduling problem involves assigning multiple jobs of different sizes to parallel production lines operating at different speeds, where the ordering of the jobs on each line is important (*i.e.*, a changeover time penalty is incurred when jobs with differing sets of inserts follow one another). More generally, suppose there are *n* zones and *m* production lines. Then the problem is equivalent to assigning the loads for each of the *n* zones to one of *m* traveling salesperson problems (TSPs) where we desire the longest optimal tour of all the TSPs to be minimized. Clearly, this can be an extremely difficult problem to solve.

OVERVIEW OF GENETIC ALGORITHMS

A GA is a heuristic search technique that mimics the theory of biological evolution to identify continually improving solutions to difficult problems (Holland, 1972). GAs have proven themselves to be an effective method for searching large, discrete solution spaces. While the solution found might not always be the global optimum, GAs typically identify a good solution in a reasonable period of time.

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are then evaluated according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem (Bergey & Ragsdale, 1999). The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes, or until no significant improvement occurs in the population for a given number of iterations.

GAs have been used successfully in a number of varied applications including newspaper distribution/delivery (Van Buer, Woodruff & Olson, 1999), energy production (Santos & Dourado, 1999) and (Shiromaru, Inuiguchi & Sakawa, 2000), vehicle routing problems (Breedam, 2000) and TSPs (Renaud, Boctor & Ouenniche, 2000). See Reeves (Reeves, 1997) for a good summary of the operation and uses of genetic algorithms.

USING A GA TO OPTIMIZE INSERT SCHEDULING

As advertisers push for micro-zoning, newspaper production operations will need to use an optimization tool to continue utilizing existing inserting equipment. To solve the insert scheduling problem using a GA, a chromosome was designed to represent the sequencing of all the loads for a particular edition of the newspaper on the available production lines. Each **load** corresponds to the newspapers and related set of inserts for a particular zone. As a simple example, suppose a particular edition of a newspaper is delivered to fifteen different zones and produced on four production lines. The chromosome shown in Figure 3-2 (explained below) represents a possible solution for this problem.

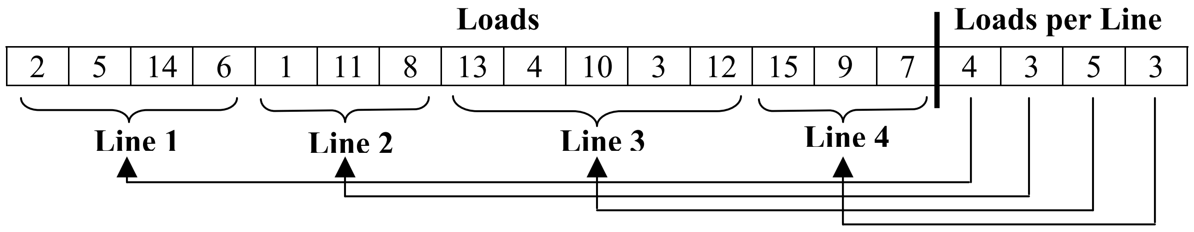


Figure 3-2
Example Chromosome Representing a Solution for 15 Zones on 4 Lines.

Note that the chromosome in Figure 3-2 consists of two-parts. The first part of the chromosome is a sequence of unique integers from one to fifteen, each identifying a specific load. The second part of the chromosome is a series of four integers indicating how the previous portion of the chromosome should be segmented to represent the four production lines. The chromosome in Figure 3-2 represents a solution where the first four loads, for zones 2, 5, 14 and 6, are processed (in that order) on the first production line, the next 3 loads, for zones 1, 11 and 8, are processed (in that order) on the second line, and so on.

So, in general, if there are *n* loads and *m* production lines the chromosome will have *n*+*m* genes. The first *n* genes are simply a permutation of the integers from 1 to *n* (sequencing the loads in the problem). The next *m* genes are a series of integers (summing to *n*) that segment the first *n* genes into *m* production lines. Note that this chromosome structure can represent any potential solution to the problem. This type of two-part chromosome is similar in structure to those used in grouping GAs (Falkenauer, 1998).

The objective for the insert scheduling problem is to minimize the amount of time required to complete all the loads for a particular edition of the newspaper. Thus, to evaluate the fitness of a chromosome we must calculate the total time required to complete the loads assigned to each of the *m* production lines. In general, the total processing time (in minutes) required to finish the loads assigned to production line *j* is given by,

$$t_j = \sum_{i=1}^{n_j} \left(\frac{Q_{ij}}{S_j} + C_{ij} \right) \qquad (1)$$

where:

n_j = number of loads assigned to line j

Q_{ij} = quantity of newspapers in the i^{th} load assigned to line j

S_j = production speed of line j (in newspapers per minute)

C_{ij} = time (in minutes) required to changeover to the i^{th} load on line j

The fitness of a particular solution is then described by the longest of the m processing times. That is,

$$\text{Fitness} = \textbf{MAX}(\, t_1, t_2, \ldots, t_{\text{m}}) \qquad (2)$$

For the insert scheduling problem, we are interested in finding a solution that minimizes the quantity in (2) (*i.e.*, a solution that minimizes the maximum processing time).

EXAMPLE PROBLEM

We examine the effectiveness of the methodology outlined above by using two popular commercial GA packages (Palisade's Evolver (Evolver, 1998) and Frontline Systems' Premium Solver (Premium Solver, 2001)) to solve the insert scheduling problem for an actual newspaper company. Both of these GA packages operate as add-ins for Microsoft Excel and cost less than $1,000, making them tools that most newspaper companies can easily afford. Both packages contain operators that can automatically generate the desired permutation of integers required for the first n genes in the solution chromosome. A simple constraint can also be specified in either package to require the values of the final m genes in the chromosome sum to the required value (n).

The data for this study was provided by a medium-size newspaper company in Southwestern Virginia that produced approximately 100,000 papers on the day being examined. The company had 53 delivery zones that were produced in three separate editions. The first edition had 9 zones with 19,000 papers, the second edition had 8 zones with 21,180 papers, and the third edition had 36 zones with 63,090 papers. The zones cannot be transferred from one edition to another because the papers for each edition are somehow unique. Thus, all of one edition must be completed prior to starting the next edition. The data represents a Wednesday edition, which is one of the heavier days for advertising inserts for this company.

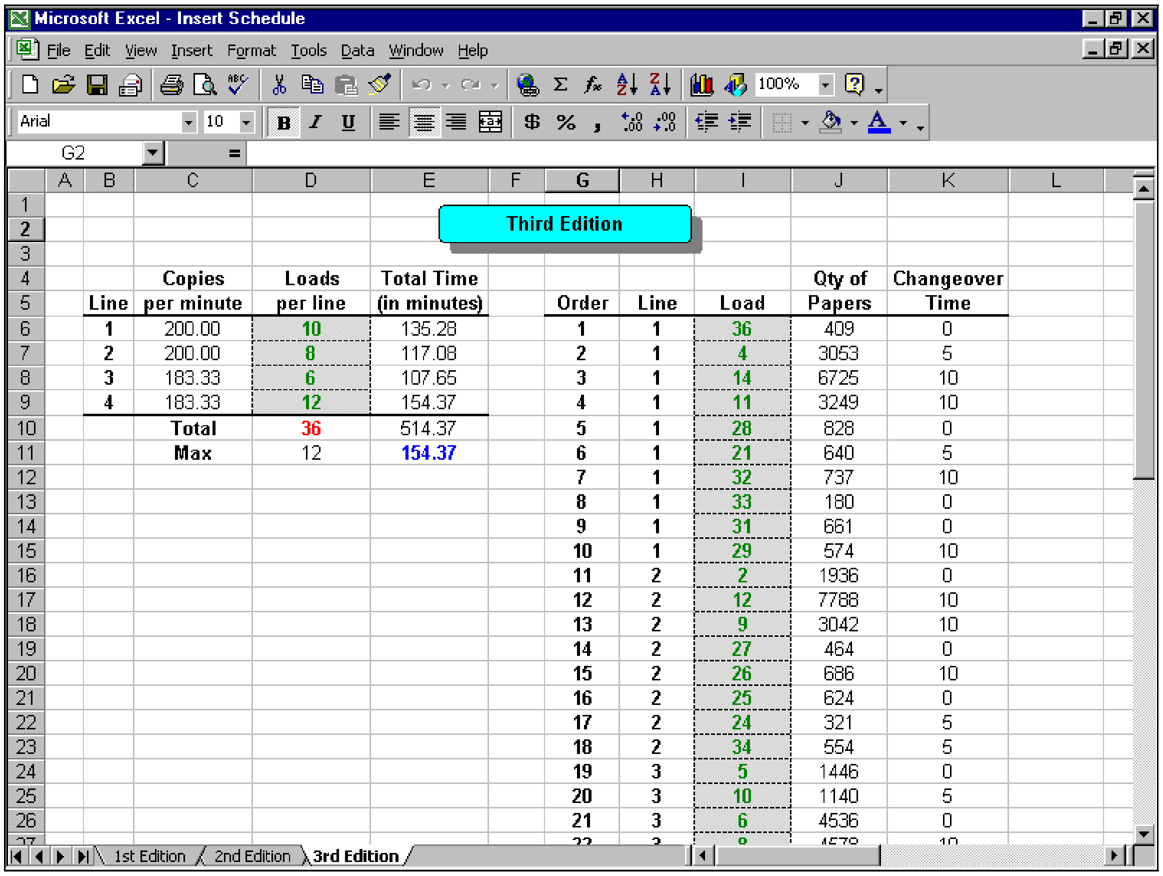
The post-press department being modeled has four insert production lines (see Figure 1) that operate at different speeds. The load for each zone is assigned to one of the production lines. The loads for some zones had identical sets of inserts, but all had a different number of papers. The zone sizes ranged from a minimum of 33 papers to a maximum of 7,788 papers.

Although the processing time model in (1) allows for variable changeover times (C_{ij}) from one load to the next, only three distinct changeover times were used in this study. Obviously, if two consecutive zones require an identical set of inserts, no changeover is required (C_{ij}=0). Otherwise, changeover times of 5 minutes or 10 minutes were used depending, respectively, on whether a minor or major change in inserts was required to move from one zone to the next on a given production line. A minor change is limited to engaging/disengaging an insert feeding position on the inserting equipment. A major change requires an insert feeding position to be restocked with a different insert and a new skid of inserts to be moved up to the machine. These changeover time estimates were provided by the newspaper's production manager and proved to be accurate when schedules were created on the spreadsheet and compared to actual production run times.

When entered into our model, the actual schedule created and implemented by the newspaper company on this day resulted in an estimated processing time of approximately 226 minutes (or 3.77 hours) to complete all three editions of the paper. In reality, it took the newspaper just over 4 hours to complete this work, with the difference in times being attributable to random equipment problems (e.g., resetting jammed inserting machines). Thus, our model seems to accurately model the actual processing times required in this operation.

THE MODEL

The spreadsheet model shown in Figure 3-3 was created to represent the insert scheduling problem for the third edition of the newspaper. (The file containing the entire model is available from the authors upon request.) Recall that this represents the largest of the three editions, with a circulation of 63,090 covering 36 zones. Similar worksheets were created for the first two editions of the paper as well.


Microsoft Excel - Insert Schedule
File Edit View Insert Format Tools Data Window Help
Arial 10 100%
G2
Third Edition
1st Edition
2nd Edition
3rd Edition

Line	Copies per minute	Loads per line	Total Time (in minutes)
1	200.00	10	135.28
2	200.00	8	117.08
3	183.33	6	107.65
4	183.33	12	154.37
	Total	36	514.37
	Max	12	154.37

Order	Line	Load	Qty of Papers	Changeover Time
1	1	36	409	0
2	1	4	3053	5
3	1	14	6725	10
4	1	11	3249	10
5	1	28	828	0
6	1	21	640	5
7	1	32	737	10
8	1	33	180	0
9	1	31	661	0
10	1	29	574	10
11	2	2	1936	0
12	2	12	7788	10
13	2	9	3042	10
14	2	27	464	0
15	2	26	686	10
16	2	25	624	0
17	2	24	321	5
18	2	34	554	5
19	3	5	1446	0
20	3	10	1140	5
21	3	6	4536	0

Figure 3-3
Sample spreadsheet for the insert scheduling problem

The values in column I of this worksheet correspond to the first n=36 genes of the chromosome presented earlier. Columns J and K contain appropriate lookup functions that reference other data in the workbook (not shown) and return the appropriate quantity of newspapers and changeover time for each load. The values in cells D6 through D9 of the worksheet correspond to the last m=4 genes of the chromosome. These values allow formulas in column H to return the appropriate line assignments. For instance, cell D6 assigns the first 10 loads to line 1, cell D7 assigns the next 8 loads to line 2, and so on.

The formula in cell D10 returns the sum of the values in cells D6 through D9 and is constrained to equal 36 in the GA packages. Cell E11 contains a function that returns the maximum of the individual total processing times for each line (computed in cells E6 through E9) and serves as the objective (or "fitness") function that the GAs attempt to minimize.

COMPUTATIONAL TESTING AND RESULTS

The production department is interested in minimizing the total amount of time required to produce all three editions. Because each edition is essentially a separate problem, the schedule for each edition was optimized individually and their individual optimal processing times were then added together to represent the total processing time.

Because GAs are a population-based heuristic search technique, the final solution reached can be influenced by the randomly generated initial starting population of solutions. As a result, we solved the insert scheduling problem associated with each edition of the newspaper 30 times to see how much variability might exist in the solution produced by the GAs.

Because the first two editions of the paper consist of very few zones and are relatively easy to schedule by hand, each GA was allowed to run for 30 seconds on each of these problems. Due to the greater complexity of the third edition, the GAs were allowed to run for a maximum of 3 minutes on this problem. Default values were used for all the various GA control settings (*e.g.*, population size, crossover strategy, and mutations rate). All computations were performed on a desktop computer with a Pentium 850 MHz processor and 192 MB of RAM. The results of this testing are summarized in Figure 3-4.

Total Processing Time
(in minutes)

	Minimum	Average	Maximum
Evolver	165.6	172.7	183.0
Solver	171.5	177.6	186.7

Figure 3-4
Estimated Processing Time Produced By GA Solutions

As indicated in Figure 3-4, both GAs produced schedules that involved significantly less processing time that the 226 minutes required by the actual schedule created and implemented by the company. Overall, Evolver outperformed Solver on this task, producing schedules that involved, on average, approximately 5 fewer minutes of processing time.

The processing times of the 30 solutions Evolver identified ranged from 165.6 to 183.0 minutes -- or 43.0 to 60.4 minutes ***less*** than that of the actual schedule implemented by the company. Clearly, the company would prefer a solution requiring 165.6 minutes over one requiring 183 minutes (though both are significantly better than the company's current schedule). To help ensure that the company obtains a "good" solution, it might implement a policy of solving the problem ***k*** times and choosing the best of the ***k*** solutions.

To test the value of this approach, we selected ***k***=4 solutions at random from the 30 solutions Evolver generated and observed the minimum processing time of these four solutions. We repeated this experiment 500 times and computed the average of the minimum of four processing times to be 168.8 minutes. Thus, if the company used Evolver to solve this scheduling problem four times (requiring approximately 16 minutes) they could expect to obtain a schedule requiring 168.8 minutes of processing time. This represents a 57.2 minute (or 25.3%) reduction in processing time from the company's original schedule and is within 3.2 minutes of the best solution shown in Figure 4.

Note that this reduction in processing time means that ***all four*** of the company's production lines would have (at least) 57 additional minutes of production capacity. This gives the company a total of 228 minutes of additional processing capacity. Because their original schedule required 226 minute of processing time, the more efficient schedule identified by Evolver basically creates capacity equivalent to one extra production line -- without actually having to buy the additional equipment for the line or hire and pay extra workers to run it.

Another benefit of the GA solution is the time required to prepare the actual schedule. The actual production schedule used typically takes in excess of an hour for an experienced person to prepare. The GA produced a schedule with only a few minutes of preparation and computational time. With an appropriate user interface developed in Excel, the results of the GA could be used directly as the actual schedule used by the production personnel.

CONCLUSIONS

Utilizing a GA can save a newspaper's post-press production operations substantial time by improving the production schedule. The improved schedule reduces the changeover time between loads and tends to balance the load between the lines. The method demonstrated in this paper reduced production time by an average of approximately 25%. This reduction in production time would result in reduced labor costs and/or allow the newspaper to accommodate more zones. The reduced cycle time in the production department would have a positive effect on two other departments within the company (news and circulation) and ultimately the final customer -- the readers. Additionally, newspaper companies that elect to move insert assembly to the day shift could also make use of this scheduling tool to enhance the efficiency of their operations and reduce labor costs.

The heuristic outlined in this paper can be applied to any problem where *n* jobs or agents must be assigned to one of *m* TSPs and we desire to minimize the maximum of the *m* optimal tour lengths. Instances of this problem class arise in a number of areas in production, logistics, and quality inspection applications. Future work will investigate the viability and effectiveness of the proposed procedure as problem size increases.

CHAPTER 4

QUALITY INSPECTION SCHEDULING FOR MULTI-UNIT SERVICE ENTERPRISES

INTRODUCTION

Many businesses in the service sector of the global economy operate with multiple physical units distributed across large geographic areas (e.g., hotels, restaurants, banks, retail stores, *etc*). To ensure a consistent level of service is delivered at units throughout the organization, these businesses often employ several regionally distributed quality inspectors who must visit and inspect each unit within his or her assigned territory on a periodic basis. Designing an effective work schedule for a quality inspector involves many of the same elements as the notoriously difficult traveling salesperson problem (TSP). However, quality inspectors typically face a number of scheduling issues that effectively impose side-constraints on the TSP formulation of the problem, creating an even more formidable scheduling challenge for multi-unit service enterprises. We refer to this challenging problem as the traveling quality inspector problem (TQIP).

Corporations using traveling quality inspectors often want the inspectors' work schedules to meet a number of goals. For instance, a unit will typically have to be inspected within a particular time window that may be based on a set frequency (e.g., two times a year), or on a variable schedule where the next inspection is scheduled based on the results of prior inspections. Additionally, corporations may want to keep trips between units within driving distance to avoid costly airfares. Fixed items on the inspector’s schedule, such as training days, vacations, and holidays, must also be taken into account. All of these items contribute to the challenge of the TQIP. Effective scheduling must take these issues (and possibly more) into account while attempting to keep the cost of the inspections as low as possible because inspection programs are operational costs that do not directly create revenue.

The hotel industry has a particular interest in increasing efficiency and reducing expenses due to the loss of business following the terrorist attacks on September 11, 2001. Since that date, the number of people traveling has decreased, resulting in a 20-50% reduction in the volume of bookings at some hotels. In order to attract customers, many hotels cut rates by 30-50% and offered other incentives such as free breakfast, discount theater tickets, and three nights for the price of two (Goodrich, 2002). The hotel industry finds itself in a business environment where the volume is dropping and, at the same time, the margin on each customer is decreasing.

In an environment where controlling expenses is of particular importance, corporate officers may view the inspection program as an area of potential savings. While reducing the frequency of inspections would reduce the costs (lower travel and labor costs), it may also result in reduced quality and a reduction in the reputation of the hotel's brand image. The resulting decrease in reputation could result in fewer bookings, or bookings at lower rates, thus lowering volume and/or margins. Another method of controlling expense (and the objective of this work) is to improve the scheduling of the inspectors so that the expense associated with the inspection program is minimized while maintaining corporate priorities. By optimizing the schedule of inspections, length of each trip, and total length of each inspector's entire tour, a company can minimize the total inspection expense while achieving the desired inspection priorities.

This paper highlights the scheduling challenges typically found in the TQIP within the hotel industry and introduces a genetic algorithm solution technique for the problem. We demonstrate the efficacy of our approach using data from an international U.S.-based hotel chain.

The remainder of the paper is organized as follows. First, we provide an overview of the issues typically encountered in a TQIP and develop a mathematical model of the problem. Next we present the specifics of an instance of the TQIP which is used to test the methodology proposed in this research and develop a framework for solving the instance of the TQIP. We then review the related work on solving the multiple traveling salesperson problem (MTSP) using genetic algorithms. Next, the methodology of a custom GA is presented as well as two methods for solving the model with a commercially available GA package. We then present a comparison of the results of the three different genetic algorithm methods used to solve the problem followed by a discussion of the benefits of the proposed method. Finally, conclusions are drawn and directions of future research are presented.

OVERVIEW OF THE TQIP

The TQIP may be viewed as a MTSP with side-constraints where, instead of a set of cities being visited by a number of salespersons, there are a set number of cities with one salesperson visiting them all in an unspecified number of short tours. While there are variations of the TSP, the Euclidean TSP is NP-hard (Schmitt & Amini, 1998; Falkenauer, 1998). The MTSP is even more difficult than the TSP because it involves determining how to optimally group the cities into tours to be assigned to the salespersons, as well as the optimal ordering of the cities within each tour.

The work on TSPs has focused on the idea of a *single* salesperson traveling in a continuous trip visiting all the cities (or units) exactly once and returning to the starting point. Another option for visiting each city exactly once is to use multiple salespersons; each making a tour of different cities such that each city is visited exactly once by one of the salespersons (a MTSP). The focus of this research is a variation on that idea; where one salesperson makes a number of short tours starting and ending at a "home" city to complete a visit to every city exactly once.

With the TQIP, every unit has some constraint associated with when it needs to be visited. Some units must be visited on a specific date (representing a hard constraint). The more common situation is that units should be visited within a given time window that may be violated if necessary (representing a soft constraint). The time window for a unit's next inspection is often based on when the last inspection was done and the score a unit received on its last inspection. A poor score would result in the unit being inspected again on an accelerated basis. While a unit may be inspected outside its time window, the goal is to perform the inspection within or as close as possible to the time window.

The process of minimizing total distance in a TSP can result in some legs of the tour being very long. This aspect of a normal TSP solution must be considered in solving the TQIP. Inspectors can only be expected to drive a certain amount of time each day and still complete an inspection of a unit. This places a limit on the distance of any single leg of the tour. If the length of a leg is too long, the inspector would have to fly to keep on schedule which tends to conflict with the ultimate goal of minimizing travel expense.

In addition to inspecting units, most quality inspectors have other duties that impact their calendars and must be accommodated in planning efficient travel schedules. For instance, inspectors often assist with providing training in hotel operations to new units in the organization. Inspectors also receive training to stay abreast of governmental regulations and industry practice. Of course, inspectors also schedule days off for vacation and to attend to personal matters. These non-inspection activities may take the form of hard or soft constraints on an inspector's schedule and must be considered in creating an effective TQIP scheduling system.

In summary, the constraints associated with the TQIP include some fixed date inspections, many variable date inspections with desired time windows, the desire to keep each leg of the tour below a certain length, and accommodating non-inspection activities on an inspector's schedule. The goal of the TQIP is to determine a schedule that minimizes the total distance traveled while satisfying these constraints as closely as possible.

MODELING THE TQIP

The TQIP may be described more formally as follows. A quality inspector travels from a home location to inspect an assigned set of P units by making N weekly tours. Let R denote the number of days the inspector will be occupied with non-inspection activities over the next N weeks (e.g., training, vacation, holidays, *etc*). Let $A = \{1, 2, \ldots, P{+}R\}$ where each integer corresponds to a location the inspector must visit to accomplish inspection and non-inspection activities over the next N weeks. Note that the same location may be visited multiple times if more than one integer refers to the same location (e.g., a unit receives training and is also inspected, or an inspector is scheduled to be at his/her home location on multiple holidays).

Let $x_{ij} \in A$ represent the location visited on the i^{th} day of tour j and n_j represent the total number of such locations visited on tour j. We assume each weekly tour j starts and ends at the inspector's home location, denoted by x_{0j}. Let d_{ij} represent the date on which location x_{ij} is scheduled to be visited and s_{ij} denote the length of stay (in number of days) required at this location. Finally, let L_{ij} and U_{ij} represent dates corresponding, respectively, to the lower and upper limits of the time window within which location x_{ij} should be visited.

The TQIP may then be stated as follows:

$$\text{MIN: } \sum_{j=1}^{N}\sum_{i=0}^{n_j} D(x_{ij}, x_{i+1,j}) + \sum_{j=1}^{N}\sum_{i=1}^{n_j}\left(w_{x_{ij}}^{+} e_{ij} + w_{x_{ij}}^{-} t_{ij}\right) + \sum_{j=1}^{N}\sum_{i=0}^{n_j} w_d M_{ij} \tag{1}$$

S.T.

$$L_{ij} \le d_{ij} + e_{ij} - t_{ij} \le U_{ij}\text{, for all } i, j \ge 1 \tag{2}$$

$$M_{ij} = \begin{cases} 1, \text{if } D(x_{ij}, x_{i+1,j}) > \theta \\ 0, \text{otherwise} \end{cases}\text{, for all } i, j \tag{3}$$

$$d_{i+1,j} = d_{i,j} + s_{i,j},\ \text{ for all } i, j \tag{4}$$

$$d_{i,j} \ne d_{k,l},\ \text{ for all } i \ne k \text{ and } j \ne l \tag{5}$$

$$C\{\, x_{ij} \mid x_{ij} = k\} = 1,\ \text{ for all } i, j \ge 1 \text{ and } k \in \mathrm{A} \tag{6}$$

$$d_{n_j,j} + s_{n_j,j} - d_{1,j} \le \beta,\text{ for all } j \tag{7}$$

$$N, n_j, e_{ij}, t_{ij} \text{ are nonnegative integers for all } i \text{ and } j \tag{8}$$

where:

β = maximum allowable number days on each tour.

$D(x_{ij}, x_{i+1,j})$ = driving distance from location x_{ij} to location $x_{i+1,j.}$

θ = maximum desired daily driving distance limit.

C = cardinality of the set.

$x_{n_j+1,j} = x_{0j}$

$w_{x_{ij}}^{+}$ = penalty for each day location x_{ij} is visited early (prior to $L_{x_{ij}}$).

$w_{x_{ij}}^{-}$ = penalty for each day location x_{ij} is visited late (after $U_{x_{ij}}$).

w_d = penalty for any leg of a tour being larger than θ.

The objective function in (1) consists of three parts. The first term represents the total distance the inspector must travel. The second term represents penalities imposed when a location is visited outside its desired time window. The slack variables e_{ij} and t_{ij} work in conjunction with constraint (2) to measure, respectively, the number of days a location is visited before or after its assigned time window. Weights of $w^{+}_{x_{ij}}$ and $w^{-}_{x_{ij}}$ are assigned to e_{ij} and t_{ij}, respectively, to penalize the scheduling of visits outside the desired time windows. Note that a fixed-date, f, for visiting a particular location $x_{ij} \in A$ may be specified by setting $L_{ij} = U_{ij} = f$ and assigning arbitrarily large values to $w^{+}_{x_{ij}}$ and $w^{-}_{x_{ij}}$. The third term in (1) operates in conjunction with constraint (3) to impose a penalty (w_d) if the travel distance between two units on a tour exceeds a maximum desired daily driving distance limit (θ).

Constraint (4) ensures that locations scheduled for a particular tour are visited on consecutive days during the tour. Obviously, it is possible for an inspector to finish an inspection on, say, Thursday and return to his/her home location Thursday evening. In this case, constraint (4) simply ensures that the inspector is scheduled to inspect a property on Thursday and be at his/her home location on Friday – it does not dictate when the inspector actually travels. Constraint (5) indicates that visits to two locations may not be scheduled on the same day.

Constraint (6) ensures that each of the activities associated with the locations specified in A are scheduled exactly once. Constraint (7) places an upper limit (β) on the number of days activities may be scheduled each week. Finally, constraint (8) specifies the variables in the problem that must be nonnegative integers.

EXAMPLE PROBLEM

This research examines an instance of the TQIP specific to a large hotel chain. The corporation uses many traveling inspectors to ensure that a guest checking into one of the chain's units is assured a certain level of service and quality. The inspectors cover such items as cleanliness, capital improvements, auxiliary services (e.g., pools, restaurants, etc.) and perform detailed inspections of randomly selected rooms. The corporation wants to improve the inspectors' schedules to ensure that each unit is inspected at least twice a year (a goal that is currently often hard to achieve) while minimizing travel expense.

Inspectors typically leave home on Monday, travel to up to 4 locations ($\beta = 4$) during the week, and then return home by Friday evening. The hotel attempts to inspect units within a time window based on when the last inspection was performed and what a unit's score was at that inspection. These inspections are an expense the corporation incurs to maintain the chain's quality and reputation.

Each inspector is responsible for all the inspections associated with a specific set of units within a specified geographical region. Each region consists of approximately 60 units. Other characteristics of the TQIP for this corporation are summarized as follows:

1. The corporation desires a maximum drive time of 6 hours (330 miles) for each leg to avoid the use of airfare ($\theta = 330$).

2. Inspectors schedule activities no more than 4 days per week.

3. The goal is to inspect each unit twice a year. After a satisfactory inspection the goal is to schedule the next inspection in a 4 to 6 month time window. Units that fail an inspection must be inspected again in a 90 to 110 day window.

4. Inspectors occasionally (approximately three times per year) perform new unit training activities that take 3 days each. These are scheduled within a 30-day window from the unit opening date.

5. Inspectors can get special inspection requests that must be done on a specific date or in a ten-day window depending on the nature of the request.

6. Each week the inspectors must submit a schedule to their supervisor for the week after next.

7. Inspectors are typically on vacation, holiday, or in training 6 to 7 weeks per year.

The data used in this study is for a single inspector responsible for a region containing 70 units ($P = 70$) in New York, Massachusetts, Pennsylvania, and New Jersey. A portion of the data for this problem is shown in Figure 4-1. The scores for each unit's last inspection are not the actual scores but, for confidentiality, were fabricated for this project by a supervisor in the inspections department and approximate actual data. All other information, such as last inspection date and unit location, is actual data for this inspector's region. A score of less than 800 requires the unit to be inspected next in a 90 to 110 day window; otherwise the unit is inspected next in 4 to 6 months. Unit names and addresses have been removed and only a number designator for each unit is being used for this study. The last inspection in the complete dataset was performed on Friday, 2/21/02, so the scheduling process starts on the following Monday, or 2/24/02.

			Window for Next Visit	
Unit No.	**Last Score**	**Last Visit**	**Beginning (L_{ij})**	**Ending (U_{ij})**
22009	850	1/1/02	5/1/02	6/30/02
22027	900	8/21/01	12/19/01	2/17/02
22042	950	7/20/01	11/17/01	1/16/02
22043	900	1/10/02	5/10/02	7/9/02
22047	850	9/27/01	1/25/02	3/26/02
31007	975	12/2/01	4/1/02	5/31/02
31012	850	2/12/02	6/12/02	8/11/02
31015	950	1/16/02	5/16/02	7/15/02
31016	990	10/11/01	2/8/02	4/9/02
31022	750	8/1/01	10/30/01	11/19/01
31028	900	1/9/02	5/9/02	7/8/02
31029	900	10/10/01	2/7/02	4/8/02
31033	975	10/30/01	2/27/02	4/28/02
31038	850	1/10/02	5/10/02	7/9/02
31044	900	9/19/01	1/17/02	3/18/02
31046	900	1/18/02	5/18/02	7/17/02
31047	900	1/14/02	5/14/02	7/13/02
31048	900	11/1/01	3/1/02	4/30/02
31049	900	10/9/01	2/6/02	4/7/02
31052	950	1/17/02	5/17/02	7/16/02
33024	950	11/6/01	3/6/02	5/5/02
33035	900	12/5/01	4/4/02	6/3/02
33054	975	1/3/02	5/3/02	7/2/02
33058	900	10/24/01	2/21/02	4/22/02
33065	850	1/2/02	5/2/02	7/1/02
33080	900	10/25/01	2/22/02	4/23/02
33086	900	11/7/01	3/7/02	5/6/02
33090	799	3/3/02	6/1/02	6/21/02
33095	900	12/12/01	4/11/02	6/10/02
33108	950	11/12/01	3/12/02	5/11/02
39009	850	2/5/02	6/5/02	8/4/02
39070	950	8/3/01	12/1/01	1/30/02

Figure 4-1
Unit Inspection Window Example

SOLUTION FRAMEWORK

To solve the TQIP model, we first created a framework within which the various units are assigned to dates in order to create an inspection schedule. The schedule is a series of tours each starting with the inspector visiting the first unit on the tour on Monday and the last unit on the tour no later than Thursday. The number of tours (N) can vary as well as the number of locations (n_j) visited in any specific tour. The framework for the schedule consists of a table with a column for each of seven data elements and a row for each date. An example of a partial schedule is shown in Figure 4-2.

Date	Day	Schedule Code	Schedule	Distances	Distance Penalty	Date Penalty
05/26/02	Sunday	X	Home	0	0	
05/27/02	Monday	H	Home	0	0	
05/28/02	Tuesday	V	Home	338.4	5000	
05/29/02	Wednesday		22043	265.6	0	
05/30/02	Thursday		39103	168.2	0	
05/31/02	Friday	X	Home	0	0	
06/01/02	Saturday	X	Home	0	0	
06/02/02	Sunday	X	Home	104.3	0	
06/03/02	Monday		33090	246.3	0	
06/04/02	Tuesday		31007	118.3	0	400
06/05/02	Wednesday		39036	102.6	0	
06/06/02	Thursday	T	Training	0	0	
06/07/02	Friday	T	Training	0	0	
06/08/02	Saturday	X	Home	0	0	
06/09/02	Sunday	X	Home	107.8	0	
06/10/02	Monday		39038	175.1	0	
06/11/02	Tuesday	N	31050	0	0	
06/12/02	Wednesday	N	31050	0	0	
06/13/02	Thursday	N	31050	209.2	0	
06/14/02	Friday	X	Home	0	0	
06/15/02	Saturday	X	Home	0	0	
06/16/02	Sunday	X	Home	280.5	0	
06/17/02	Monday		33035	421.1	5000	1400
06/18/02	Tuesday		39009	137.8	0	
06/19/02	Wednesday	F	31038	126.6	0	
06/20/02	Thursday		39097	110.7	0	
06/21/02	Friday	X	Home	0	0	
06/22/02	Saturday	X	Home	0	0	

Figure 4-2
Model for TQIP Instance

The first two columns in Figure 4-2 indicate the date and day being scheduled, respectively. The "Schedule Code" column indicates if the specific date is assigned for a fixed unit inspection (F), vacation (V), holiday (H), training day (T), a day at home (X) or a three-day new property training session (N). Any date without a code in the schedule code column is an open date that can be used to inspect a unit. The methodology of this research is based on the premise that any conflicting hard constraints (fixed date assignments) will be resolved by human intervention prior to creating a schedule with the proposed heuristic. For instance, if a unit had a fixed inspection on day the inspector was schedule for vacation the inspector would have to resolve this conflict prior to creating a schedule with this tool (by moving either the vacation or inspection to another day).

The "Schedule" column in Figure 4-2 indicates where the inspector is scheduled to be on each specific date. The "Distance" column indicates the distance between the unit in each row i and the unit in row $i+1$ immediately following it ($D(x_{i,j}, x_{i+1,j})$). The distance values are looked up from a matrix (not shown) containing the driving distances between every unit (including the inspector's home). The "Distance Penalty" column shown in the second to last column indicates the penalty incurred if a specific trip is over a set distance. For the purpose of this problem the penalty was set as w_3 = 5000 miles for any trip over 330 miles (approximately a 6 hour drive). The values for the penalties were based on conversations with an inspection supervisor and the importance of minimizing airfare versus the desired for units to be inspected on time.

Finally, the "Date Penalty" column in Figure 4-2 assigns a penalty ($w_1 e_{ij}$ or $w_2 t_{ij}$) if the unit on that row is scheduled for inspection outside its date window. For this research any unit assigned a date outside the desired window is assessed a penalty of 100 miles per day outside the window ($w_1 = w_2 = 100$) . The sum of the final three columns determines the fitness of the given schedule.

OVERVIEW OF GENETIC ALGORITHMS

Because of the difficulty in solving the TQIP, heuristics must be used in order to obtain a solution in a timely fashion. Some possible heuristics available include neural networks, tabu search and genetic algorithms. Due to the similarity of the TQIP to the MTSP, we have selected genetic algorithms based on the successful application of GAs to the MTSP (Malmborg, 1996 & Park, 2001).

A GA is a heuristic search technique that mimics the theory of biological evolution to identify continually improving solutions to difficult problems (Holland, 1975 & Holland, 1992). GAs have proven themselves to be an effective method for searching large, discrete solution spaces. While the solution found might not always be the global optimum, GAs typically identify a good solution in a reasonable period of time.

In a nutshell, GAs work by generating a population of numeric vectors (called chromosomes), each representing a possible solution to a problem. The individual components (numeric values) within a chromosome are called genes. New chromosomes are created by crossover (the probabilistic exchange of values between vectors) or mutation (the random replacement of values in a vector). Mutation provides randomness within the chromosomes to increase coverage of the search space and help prevent premature convergence on a local optimum. Chromosomes are then evaluated according to a fitness (or objective) function, with the fittest surviving and the less fit being eliminated. The result is a gene pool that evolves over time to produce better and better solutions to a problem (Ragsdale, 2001). The GA's search process typically continues until a pre-specified fitness value is reached, a set amount of computing time passes, or until no significant improvement occurs in the population for a given number of iterations.

Solving TSPs using GAs has generated a great deal of research focused on how best to perform the action of “evolving” an optimal (or good) solution to the problem. The operators used in the GA are critical to the ability of a GA to find a good solution to the problem. The TSP is unique in that it requires the solution to have each city visited once and only once (with no sub-tours). This requires the operators to perform crossover and generate the new child solutions while maintaining feasibility (each city is visited once and only once).

A number of different crossover methods have been proposed in the literature to solve the TSP using a GA. Some of the most commonly used operators include: Order Crossover (Goldberg, 1989; Davis, 1985; Starkweather, Whitley, Whitley, & Mathial, 1991; Oliver, Smith & Holland, 1987), Partially Mapped Crossover (Goldberg, 1989; Starkweather, et al, 1991; Goldberg and Lingle, 1985; Oliver, Smith & Holland, 1987), Cycle Crossover (Goldberg, 1989; Starkweather, et al, 1991; Oliver, Smith & Holland, 1987) and Asexual Crossover (Chatterjee, et al, 1996; Schmitt and Amini, 1998; Fox & McMahon, 1991). Other TSP operators that have been proposed include utilizing a matrix chromosome representation (Poon and Carter, 1995; Qu and Sun, 1999), hybrid operators (Laiw, 2000), simple crossover (Knosala & Wal, 2001) ordered crossover #2 (Syswerda, 1989), moon crossover (Moon, Kim, Choi & Seo, 2002) and position based crossover (Syswerda, 1989).

MTSPs have been modeled and solved with GAs by those solving vehicle scheduling problems (Malmborg, 1996; Park, 2001). The vehicle scheduling problem (VSP) consists of scheduling a fleet of vehicles to visit *P* locations with each city being visited by one and only one truck. The VSP faces constraints on the number of locations each truck can visit due to the capacity of each truck available and the size load to be picked up at each location. In some cases, the locations must be visited between specific times called time windows. A variety of objectives can be considered: minimum total distance, minimum number of trucks required or minimum lateness (if time windows are used). These issues lead to a number of different possible configurations for the VSP: VSP with/without time windows, VSP with heterogeneous/homogeneous truck capacities, and VSP for minimum distance/minimum truck requirements. If modeled properly the MTSP can be solved using any of the GA operators developed for the TSP.

A CUSTOM GA FOR THE TQIP

To solve the TQIP, we developed a custom GA called Best-Trip. The program was written in VBA and uses Excel for a user interface. The details of the program are described below.

Chromosome Design

To solve the TQIP using a GA, we must first design an appropriate chromosome for the problem. It is possible for the schedule displayed in column 4 of Figure 4-2 to be encoded and used as a chromosome in the population for the GA. However, due to the function of crossover and mutation, many of the offspring created using this chromosome would be infeasible and have to be corrected for the fixed dates following the GA operations (i.e., to restore the fixed dates and weekends to the correct dates). In order to avoid this dilemma, the chromosome used by Best-Trip is simply a permutation of integers representing units and activities that are not assigned to fixed dates. The permutation of units in the chromosome is created from the schedule. The chromosome is created by starting at the beginning of the schedule and adding any unit number to the chromosome which does not have a value in the Schedule Code column (See Figure 4-2).

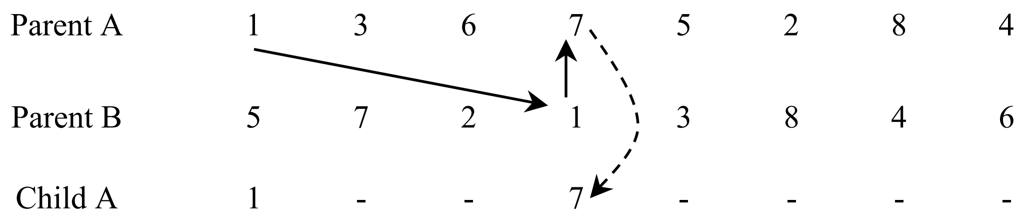
Evolutionary Operators

Best-Trip utilizes cyclic crossover with a roulette wheel selection methodology (Goldberg, 1989). Cyclic crossover is a commonly used TSP operator and is described in detail below. Roulette wheel selection first assigns each population member a probability of being selected based on its fitness value. A chromosome with a better fitness value has a higher probability of being selected. Two parents are then randomly selected from the population based on the probabilities assigned to each member of the population. This can be viewed as each member of the population having a space on a roulette wheel proportional to its fitness value and then spinning the wheel to select the parents.

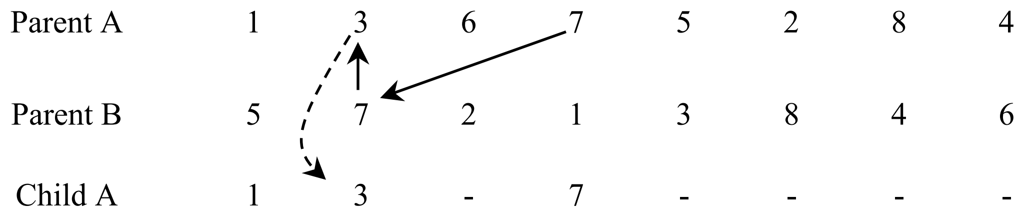
Cyclic crossover (Goldberg, 1989) was chosen due to the fact that it creates child solutions that have the value from one of the two parents in each position in the child chromosome. This property of CX was considered important to the Best-Trip algorithm as the initial population is seeded with desirable traits (as explained below) that relate to the units absolute position in the chromosome. Cyclic crossover starts with the value in the first position of the first parent chromosome. Based on that gene's value, a cycle through the parent chromosomes is used to populate two child solutions. For example, cyclic crossover (CX) starts with the first gene of the first parent and places that gene in the first position of the first child solution.

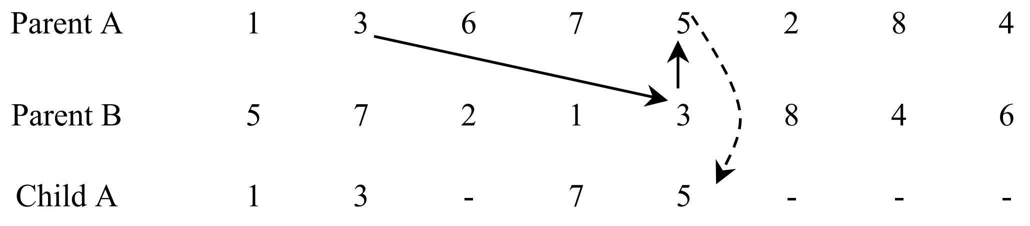
Parent A	1	3	6	7	5	2	8	4
Parent B	5	7	2	1	3	8	4	6
Child A	1	-	-	-	-	-	-	-

Child A will be a combination of the two parent with each position being filled with a value from the corresponding position in one of the parents, therefore the value of the first position of parent A (in this case 1) cannot be used from parent B (in this case the fourth position of parent B). So to maintain a feasible solution, the value of the fourth position must be taken from parent A (since taking that value from parent B would result in two 1s in child A). Therefore a second gene is set based on the value of the first gene and the need to maintain a feasible TSP solution. The position of the value of the first gene in child A (1) is found in parent B (the fourth position in parent B). The value in that position of parent A (7) is then set as the value for the corresponding position of child A.



Now the next gene can be set based on the value of the gene in the position just set in child A. That value is found in parent B and its position is noted to set the next value in child A as the value in that position of parent A. This cyclic procedure is continued until the value set in child A matches the value in the first position of parent B.





Now the positions in child B corresponding to the positions set in child A are taken from parent B.

Child B	5	7	-	1	3	-	-	-

The empty positions in the child solutions are then filled in with the values from the other parent's chromosomes.

Child A	1	3	2	7	5	8	4	6
Chile B	5	7	6	1	3	2	8	4

Fitness Evaluation

After the completion of the crossover operator, each child chromosome is placed into the schedule framework (see Figure 4-2) to allow calculation of the new chromosome's fitness. The procedure for turning the chromosome into a schedule starts by selecting the unit in the first gene of the chromosome and assigning it to the first open period of adequate length in the schedule after the starting date of the unit's time window. Once that unit is assigned, the unit in the next gene is selected and assigned to the first open period of adequate length in the schedule that is after the start date of the unit's window and after the completion of the previous unit.

The assignment process is continued until all the units in the chromosome are assigned dates on the schedule. The process takes the chromosome and creates a schedule placing all of the units on the schedule in the same relative order they appear in the chromosome and ensuring they will not be scheduled before the first date of their time window.

Next, open days in the schedule are compressed and the fitness calculated. Converting the chromosome into a schedule can leave open days within the week. An example would be when there is a unit on Monday, nothing on Tuesday and units on Wednesday through Thursday. The compression routine would remove the open day on Tuesday by moving the units on Wednesday through Thursday up one day and assigning Thursday as being a day at home. This creates a continuous trip for each week with any unassigned days (when the inspector would be at home) being grouped in a weekend.

Once the compression of open days is completed, the fitness of the solution is calculated by summing the total driving distance, distance penalties and date penalties. If the child solution has a better fitness (lower total distance with penalties) than its parent, it replaces its parent in the population.

Initial Population Generation

Another important feature of the Best-Trip is the manner in which the initial population is created. The creation of the initial population is done in such a way as to ensure variability in the population while creating a good starting point for the program. Each solution begins as an empty schedule (with only the fixed dates on the schedule). Next, an unassigned unit is randomly selected and placed on the schedule in the first available period of adequate length occurring at least five days after the first date of its time window (L_{ij}). The unit is placed at least five days after the first date of the time window to ensure that during the compression procedure the unit is not moved from being in its time window to being before its time window (and thus incurring a penalty). This procedure is repeated until all the units have been selected and placed on the schedule.

Once all the units are on the schedule, any open days in the schedule are compressed and the fitness of the schedule is calculated. The schedule is then reduced to a chromosome by eliminating fixed date activities from the schedule and the new chromosome is added to the population. This procedure is repeated until the entire initial population is created. By using this heuristic to create the initial population, the activities with time windows are approximately within the right time window. The procedure provided no means to place units such that either total distance was minimized or distance between any two units was less than 6 hours driving time.

METHODOLOGY

To assess the effectiveness of the Best-Trip algorithm on the TQIP, we compared its performance against the commercial GA program Evolver (Evolver, 1998). Specifically, we tested three solutions techniques described as follows:

1) **Best-Trip** – The custom GA described in this paper.

2) **EV-Date** – Evolver, using a date priority for creating a solution from the chromosome.

3) **EV-Sequence** – Evolver, using a sequence priority for creating a solution from the chromosome.

The EV-Date technique uses a date priority method for creating a schedule from the chromosomes. The date priority heuristic starts with the first unit in the chromosome and schedules it for the first open period of adequate length after the starting date of the unit's time window. Next, the second unit in the chromosome is scheduled in the first open period of adequate length after the start of its time window. This process is continued until all of the units in the chromosome are scheduled. When using the date priority the resulting schedule will likely result in the units appearing in a different order on the schedule than they appear in the chromosome.

The EV-Sequence technique uses a sequence priority for scheduling the units from the chromosome. The sequence priority heuristic starts with the first unit in the chromosome and schedules it in the first open period of adequate length after the beginning of the unit's time window. The next unit in the chromosome is scheduled after its time window's starting date and also after the date the pervious unit was scheduled. This process is continued until all the units in the chromosome are scheduled. By using this method the units appear in the same order in the schedule as they appear in the chromosome. This is the same method used by Best-Trip to create a schedule from the chromosome when the custom GA method was used.

An example of how the first four units in a chromosome would be scheduled by each method is shown in Table 4-1.

Units order in Chromosome
31015
31047
31046
31052

	Window for Next Visit	
Unit No.	**Beginning**	**Ending**
31047	5/14/02	7/13/02
31015	5/16/02	7/15/02
31052	5/17/02	7/16/02
31046	5/18/02	7/17/02

Using Date Priority for Creating Schedule				**Using Sequence Priority for Creating Schedule**			
5/13/02	Monday			5/13/02	Monday		
5/14/02	Tuesday		31047	5/14/02	Tuesday		
5/15/02	Wednesday			5/15/02	Wednesday		
5/16/02	Thursday		31015	5/16/02	Thursday		31015
5/17/02	Friday	X	Home	5/17/02	Friday	X	Home
5/18/02	Saturday	X	Home	5/18/02	Saturday	X	Home
5/19/02	Sunday	X	Home	5/19/02	Sunday	X	Home
5/20/02	Monday		31052	5/20/02	Monday		31047
5/21/02	Tuesday		31046	5/21/02	Tuesday		31046
5/22/02	Wednesday			5/22/02	Wednesday		31052

Table 4-1
Sample of Schedules Created Using Date and Sequence Priority Methods

For all three methods, a population of 100 was used with the stopping criteria of 15 minutes of run time. Mutation was not used in Best-Trip as the disruptive nature of mutation was not expected to improve the population due to the structure of the chromosomes. When using Evolver, all control parameters were left at the default settings (expect for population being set at 100 and stopping criteria being set at 15 minutes).

COMPUTATIONAL TESTING AND RESULTS

Because GAs are heuristic in nature and involve some randomness in the changes to the chromosome, the solution obtained can vary with each run of the problem. In order to assess the quality and variability of the solutions produced by the GA tools, each of the three methods was used to solve the problem 30 times. Each run was for 15 minutes and was carried out on a desktop machine using a Pentium 4, 1.6 GHz processor with 256 MB of RAM. Summary results for these runs are displayed in Figure 4-3.

The values shown in Figure 4-3 summarize the minimum, maximum and average fitness function values for each solution method. The Best-Trip method produced the lowest minimum, average and maximum values of the three techniques. In fact, the worst schedule produced with the Best-Trip method had a lower fitness then the best schedule produced by either EV-Date or EV-Sequence. Between the two Evolver methods, the EV-Date was superior to EV-Sequence.

	Fitness Function Value (Distance with Penalties in miles)		
Solution Method	**Minimum (miles)**	**Average (miles)**	**Maximum (miles)**
EV-Sequence	97,709	127,543	157,409
EV-Date	78,382	99,902	124,319
Best-Trip	62,024	68,324	74,740

Figure 4-3
Summary Computational Results

The fitness function is the sum of three components; the driving distance, date penalty and distance penalty. A breakdown of the components of the fitness function for each of the three GA tools used is displayed in Figure 4-4 (Min, Ave and Max values are based on Fitness Value).

		Distance (miles)	**Date Penalty (miles)**	**Distance Penalty (miles)**	**Fitness Value (miles)**
Best-Fit	Min	12,224	49,800	0	62,024
	Ave	12,691	54,800	833	68,324
	Max	13,540	56,200	5,000	74,740
EV-Date	Min	13,482	64,900	0	78,382
	Ave	14,055	80,847	5,000	99,902
	Max	14,719	94,600	15,000	124,319
EV-Sequence	Min	13,009	79,700	5,000	97,709
	Ave	13,237	101,973	12,333	127,543
	Max	13,109	129,300	15,000	157,409

Figure 4-4
Breakdown of Fitness Function Values

The computational results clearly show Best-Trip performs statistically better than either EV-Date or EV-Sequence (*p*-value=1.9E-16 and 9.8E-19 respectively) at solving the model developed. The solutions created with Best-Trip would generally drive all of the distance penalties out of the solution while the Evolver solutions often left some legs of the tours long enough that they incurred a distance penalty. The superiority of Best-Trip can likely be attributed to the nature of the CX operator. With CX, each position in the child chromosomes is filled from the corresponding position of one of the two parent chromosomes. This results in less disruption of the tours as the operator creates new solutions.

As Evolver is a proprietary software package, the GA operators it uses are not known. Evolver may use a crossover method that is more disruptive to the parent solutions than CX. However, it almost certainly does not use a heuristic to seed the initial population with desirable traits. Additionally, the TQIP is highly constrained as every unit is assigned a fixed date or has a soft constraint. The soft constraints create a linear increase in the penalty as the unit moves away from the desired inspection window. With such a heavily constrained problem, if a disruptive operator is used on a good solution, it is likely to disrupt the parts of the chromosome that make the solution desirable.

Sensitivity Analysis of Penalties

The penalties associated with long trip distances and units being inspected outside the desired date windows impact the total distance of the tours. As a result, management must determine the value of enforcing certain limits on date and driving distance issues versus the total cost of the tour due to actual driving miles. In order to investigate the effect these penalties have on driving distance (and therefore expense), additional runs were conducted using the custom GA. Thirty runs were conducted with various date/distance penalty combinations to determine the effects of these penalties.

The first scenario run assumes management is unconcerned with long drives but wants properties inspected on time. Setting the date penalty = 100 and the distance penalty = 0 simulated this scenario. The averages of the thirty runs for each scenario are presented in Figure 4-5 (Note: the Min, Ave and Max values presented in Figure 4-5 are based on the fitness value.)

		Distance (miles)	**Date Penalty (miles)**	**Distance Penalty (miles)**	**Fitness Value (miles)**
Date Penalty = 100 Dist. Penalty = 0 (Min Window Violations)	Min	12,189	45,300	0	57,489
	Ave	13,031	46,523	0	59,554
	Max	13,325	49,500	0	62,825

Figure 4-5
Sensitivity Analysis Results - Minimizing Window Violations

Setting the distance penalty = 0 and leaving the date penalty = 100 had little effect on the driving distance. The min and max values both decreased slightly (35 and 215, respectively), while the average distance actually increased (340 miles). The effect that the distance penalty has can actually help some solutions due to the fact that it drives long trips out of the schedule, tending to reduce the driving distance. While the reduction in trip length would generally reduce the driving distance, there are cases where it increases the driving distance due to a long trip length being necessary to reach the optimum (or better) solution. This causes in variation in the results depending on if the long trip length driven out of the solution was indeed necessary to achieve a lower total distance or just a long trip which when removed reduced the overall driving distance.

The next scenario assumes management is interested in preventing long driving trips, but is not going to enforce date windows. This was simulated by setting the date penalty = 0 and a distance penalty = 5000. The averages of the thirty runs for each scenario are presented in Figure 4-6 (Note: the Min, Ave and Max values presented in Figure 4-6 are based on the fitness value.)

		Distance (miles)	**Date Penalty (miles)**	**Distance Penalty (miles)**	**Fitness Value (miles)**
Date Penalty = 0 Dist. Penalty = 5000 (Min Long Trips)	Min	10,656	0	0	10,656
	Ave	11,284	0	167	11,451
	Max	12,106	0	0	12,106

Figure 4-6
Sensitivity Analysis Results - Minimizing Long Trips

Setting the date penalty = 0 and leaving the distance penalty = 5000 did result in a significant reduction in the driving distance. The min, average and max values observed all showed a reduction (1568, 1407 and 1434, respectively). These reductions in distance were statistically significant (*p*-value of 1.804E-12). Clearly trying to force the properties into certain time windows has a significant impact on the distances the inspectors must drive.

The final scenario investigated assumes that management wants to minimize total driving distance and was unconcerned with long trips or properties being inspected in the correct time windows. We simulated this by setting both the date and distance penalties to zero. The averages of the thirty runs for each scenario are presented in Figure 4-7 (Note: the Min, Ave and Max values presented in Figure 4-7 are based on the fitness value.)

		Distance (miles)	**Date Penalty (miles)**	**Distance Penalty (miles)**	**Fitness Value (miles)**
Date Penalty = 0 Dist. Penalty = 0 (Min Distance)	Min	9,809	0	0	9,809
	Ave	10,297	0	0	10,297
	Max	10,944	0	0	10,944

Figure 4-7
Sensitivity Analysis Results - Minimizing Distance

The elimination of all penalties and simply trying to reduce the total driving distance also had a significant impact on the driving distance. The min, average and max were reduced 2416, 2394 and 2596, respectively (from the distance penalty = 5000 and date penalty = 100 runs). This was again a statistically significant reduction in the driving distance (*p*-value = 4.348E-20).

An interesting outcome is the different effects that setting the distance penalty to zero had on driving distance relative to the value of the date penalty. With a date penalty of 100, there was no significant difference in the results with or without a distance penalty. But with a date penalty of 0, there is a significant difference between using a distance penalty of 5000 and the distance penalty of 0. With a date penalty of 0, the difference between a distance penalty of 5000 and a distance penalty of 0 resulted in a reduction in min, average and max values of 847, 987 and 1162, respectively. This was a statistically significant reduction in distance (p-value of 3.778E-11). In other words, removing the distance penalty when there was a date penalty had no effect on driving distance, whereas removing the distance penalty when there was no date penalty had an effect on driving distance. This difference can be explained by the fact that when there is no date penalty, there are in effect more solutions that can be explored in order to try and find the optimum solution. Clearly, the effect the date penalty has is significant in any case as it encourages the schedule to meet certain scheduling criteria which have a detrimental impact on the goal of minimizing total driving distance.

Longitudinal Analysis

Next, five simulation runs were conducted to determine the long-term effects of using the scheduling tool over time. The simulation runs were based on the original data supplied and included the following assumption:

1) Inspections were scheduled for Monday through Thursday with Friday being scheduled at home.

2) Inspectors were scheduled for a total of 6 weeks of vacation, holiday and training.

3) Each unit had a 5% chance of failing an inspection.

4) Three new property training sessions were scheduled in the twelve months and a total of 20 special inspections were conducted throughout the twelve months.

Each simulation run consisted of creating a new schedule at the beginning of every month for 12 months to simulate the effects of using the tool for a year. The procedure for the simulation was as follows: a schedule was produced, the inspector "completed" one month of inspections, the inspected units' data for last inspection date and score were updated, a new schedule was created for the following month and data was collected on the new schedule. The data collected for each month included the number of units currently overdue on the first day of the schedule, the total driving distance of the schedule developed, the distance penalty, the date penalty and the total fitness of the schedule created.

The Results of Using Best-Trip

The data collected shows the effects of the scheduling tool over time are summarized in Figure 4-8. By using Best-Trip, the number of units late (past the last day of their inspection time window) at the beginning of each new schedule is reduced from a starting value of 9 to almost 0 within 5 months. Continued use of Best-Trip keeps this number near zero. As can be expected, with the reduction of the number of units currently late, the date penalty also goes down during the first 5 months. After the fifth month, it can be expected that the number of units currently late will average less than 1. This supports the goal of inspecting each unit on time and twice a year (time windows for passing units was 4-6 months).

Month	1	2	3	4	5	6
Number of Late Units	9	6.2	4.2	0.6	0.2	0.4
Driving Distance	12,084	12,311	12,705	12,518	12,956	13,362
Distance Penalty	0	1,000	4,000	1,000	0	1,000
Date Penalty	65,260	46,280	22,900	9,920	3,340	4,160
Total Fitness	77,344	59,590	39,603	23,438	16,296	18,522

Month	7	8	9	10	11	12
Number of Late Units	0.2	0.2	0.2	0.2	0	0.6
Driving Distance	13,605	13,173	13,889	14,303	13,449	12,804
Distance Penalty	1,000	2,000	5,000	7,000	2,000	5,000
Date Penalty	5,720	3,480	4,760	8,280	1,900	6,400
Total Fitness	20,325	18,653	23,649	29,583	17,349	24,204

Figure 4-8
Simulation Results

The long-term effects on late properties and driving distance using Best-Trip can be seen in Figure 4-9. The graph clearly shows that late properties will be caught up and kept at a low value with Best-Trip. On the other hand, driving distance actually trends upward and increases over time (at a statistically significant rate). A possible explanation for this somewhat unexpected anomaly is that with fewer late units, the viable solution space increases in size. The increase can result in additional local optimum or just require a longer run time to reach a good solution. With an increase over time and a reduction in late units, management may need to carefully consider the value at which the date penalty should be set (which our earlier sensitivity analysis showed had the greatest effect on driving distance).

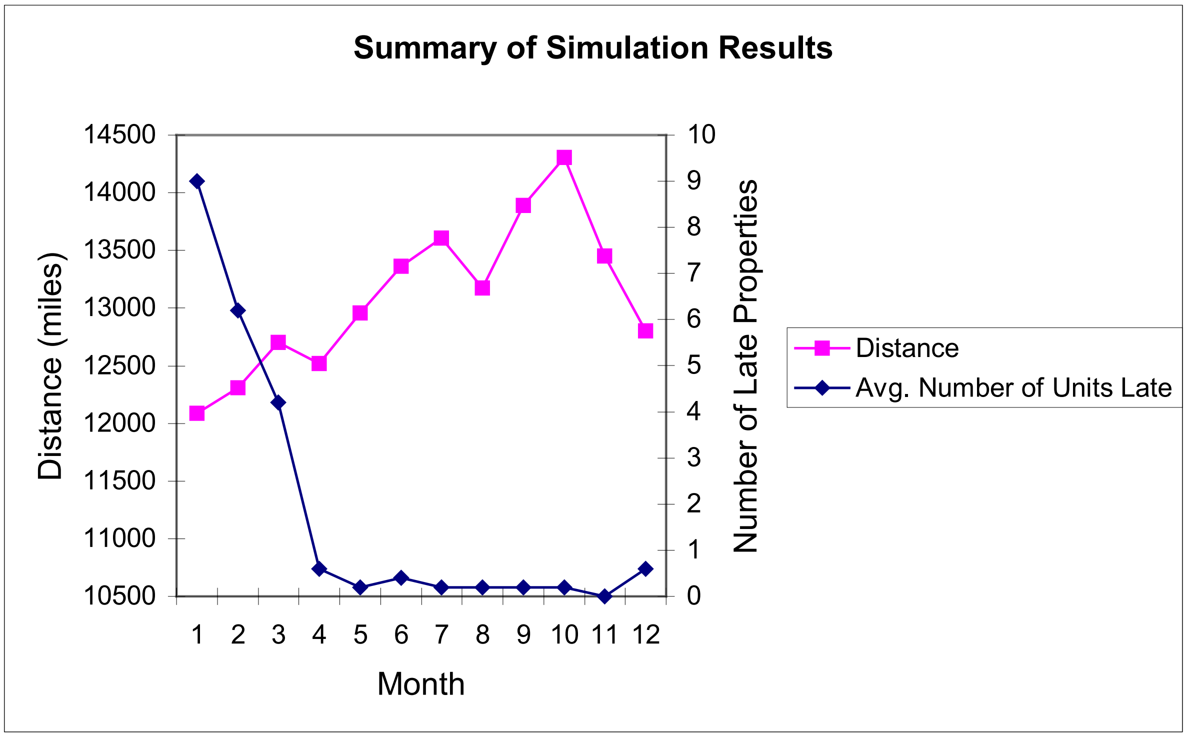


FIGURE 4-9
Graphical Display of Sensitivity Analysis

One option management has is once the units inspections are all current, the date penalty may be reduced to facilitate a reduction in the driving distance. With a flexible tool like the one proposed, the date penalty can be altered month to month depending on the number of units currently overdue for inspection.

The final point to consider from the simulation results is the number of units. The region modeled is currently assigned 70 units. The corporate goal is currently to have each region made up of 60 units. The results of the simulation indicate it is possible for an inspector to handle 70 units while keeping the units on a twice a year inspection schedule. Having each inspector handle an additional 10 units could reduce the number of inspectors required and result in a significant labor savings.

BENEFITS OF PROPOSED METHODOLOGY

Solving the TQIP with the Best-Trip technique provides numerous benefits to both corporate management and the inspectors who ultimately have to adhere to the schedule produced. The corporate offices gain additional oversight of reimbursement expenses and the ability to have a greater direct impact on the priorities that will be set in producing a schedule. For the inspectors, benefits include reduced effort in producing a schedule, assurance of an effective schedule even if they are inexperienced at producing such a schedule, and an assurance that their schedules are in accordance with corporate priorities.

Best-Trip will also make certain financial aspects of the inspection program easier to predict and monitor. The tool will produce a schedule for every inspector for the next four to six months. While the part of the schedule past the three-month mark is very likely to change, the schedules for the next several months should be relatively accurate. With this data from all of the inspectors, the corporate office can project expenses with greater accuracy. Another benefit of this tool is that it gives management a benchmark against which to check inspector mileage charges to ensure inspectors are not creating longer trips to run up unnecessary mileage in order to receive larger reimbursement payments.

Management gains the ability to directly control, set and change the priorities that will be used to determine inspector schedules. Management can use the penalty function values to change the priorities that will be placed on keeping units within the desired time windows and keeping each leg of the trip under a certain distance. Not only can the penalty values be changed, thus refocusing priorities, but all the inspectors in the corporation can also use the same values. Management will gain the ability to set different priorities in different regions due to the geographical or business conditions which apply. The priorities can be changed and implemented at will just by setting new penalties and rerunning the schedule.

When packaged in the form of a DSS, the proposed methodology makes it easy for even inexperienced inspectors to produce a good schedule. Clearly the ability of an inspector to produce a good schedule will be partially a function of his experience at creating schedules and his knowledge of the units and areas he is visiting. The tool we have produced will make it possible for inspectors who have not yet visited the first unit in their area to generate a good schedule. The schedule produced in also going to be in alignment with corporate goals because management will set the penalties.

The tool will also produce alternatives for the inspectors to choose from. This can be accomplished by one of two ways. First, by the nature of genetic algorithms there will by many solutions to the problem created. Many of these solutions will likely be almost as good as the best solution found. If the inspector would like to see other good options they would be readily available within the population used to create the suggested best solution. The other method would be to set the program up to perform multiple runs and display the best of each run.

The instance of the TQIP used in this research is a multiple objective optimization problem. The multiple objectives have been handled in this research by setting values for each objective (distance and date penalties) in terms that can be summed to create a single fitness value for each solution. If management wanted avoid the use of penalties and optimize all the objectives equally, then a Pareto ranking could be used. Pareto rankings have been successfully used with GA (Goldberg, 1989) and can be used to evaluate the fitness value of the chromosomes. A Pareto ranking system would eliminate the need for management to select specific values for each penalty to solve the scheduling problem.

CONCLUSIONS

This research describes the TQIP and proposes an effective solution technique for this difficult problem. A mathematical model was developed and a GA optimization tool developed which effectively solved the scheduling problem. The custom program called Best-Trip was written to solve the problem and produced results superior to a commercially available GA package. The custom program consistently drove the distance penalties out of the solutions and worked to minimize the date penalties while minimizing the total distance traveled by the inspector on his tour of the units.

Best-Trip was demonstrated with a specific instance of the TQIP and results in a number of benefits to both corporate management and the inspectors who must make the tours. Management gains the benefit of being able to quantitatively set certain priorities while minimizing total travel expenses. The program can also be used by managers to predict near term budget expenses as well as monitor the actual travel expenditures of individual inspectors. From the inspector's perspective, the gains include reducing the time spent creating a schedule every week and ensuring that his personal schedule meshes with corporate priorities.

Areas of future interest include other instances of the TQIP and of the MTSP. There are a number of production (multi-line product planning) and scheduling problems (many variations of the vehicle scheduling) that can be modeled as a MTSP and may benefit from the use of a scheduling tool similar to the one developed for the TQIP. Additionally, there may be other methods of optimizing or improving the model developed in this research by using other optimization methods or modifying the proposed GA tool.

Chapter 5

Summary and Directions for Future Research

SUMMARY

The MTSP is a difficult problem for which we have developed, tested and implemented a new two-part chromosome for use with GAs. The new chromosome is superior to existing chromosome representations due to its smaller solution space with fewer redundant solutions. Computational testing comparing the two-part chromosome to the two existing chromosomes showed the two-part method's advantages.

This research has presented a theoretical comparison of two existing genetic algorithm chromosome representations to solve the MTSP to the new two-part chromosome representation. Included in this research has been the application of the new chromosome method to a production scheduling problem we have encountered. While the problem solved is a beginning, there are numerous other areas to be researched.

Additionally, a traveling quality inspector problem was modeled as a multiple traveling salesperson problem and a custom GA was presented to solve the problem. Further research into other applications which can benefit from such an algorithm and methods of improving the algorithm are warranted.

FUTURE RESEARCH

Vehicle Scheduling Problem

One of the most active research areas to date using a MTSP model has been the vehicle scheduling problem (VSP). While there has been a number of applications of genetic algorithms to solve the VSP, none has used the new chromosome representation presented in this research. The nature of the VSP makes it a challenging problem due to the many constraints (and variations thereof) which are possible. Each of these major variations in the problem will be areas to be researched to determine the if the benefits of the new chromosome representation will provide improvements over existing techniques. The major variations include:

Time Windows. The VSP with time windows has a constraint on some (or all) of the sites to be visited that they must be visited within a specific time window. This specific variation on the VSP is particularly applicable for companies involved in production using just-in-time scheduling. The time window constraint also applies to many parts of the trucking industry where pick up and drop off of loads must be within specific shipping and receiving hours.

Pick Up/Drop Off Precedents. Another variation of the VSP involves the addition of constraints that place sequencing requirements on the pick up and drop off of the loads. This means that a certain site must be visited before another site and both sites would have to be visited by the same salesperson. This problem is typical of the requirements faced by courier services.

Minimum Trucks/Minimum Distance. The VSP problem is often solved using either of the two objectives: minimizing the number of trucks, or minimizing driving distance. There are times when both of these objectives are to be considered. Multiple objective approaches to the VSP have generally followed goal programming techniques, assigning a relative importance weighting factor to each of the objectives. This problem warrants additional research using a Pareto ranking to exploit the population-based solution approach of the GA.

DSS Development for Salesperson/Inspectors

This research successfully applied a custom GA using an MTSP model to the difficult traveling quality inspector problem (TQIP). While the custom GA could be used to develop a schedule for the problem, it lacked the functionality that an effective decision support system would provide. A DSS tool should be developed for use by management and inspectors for solving this difficult scheduling problem to complete the research on this instance of the TQIP.

New or Improved Operators for the MTSP

The operators used in this research for solving the MTSP were only a few of the possible options available. While the operators used were successful, the research was clearly not exhaustive in this area. Additional research into the other available operators should be conducted to investigate the possibility that a different operator exists that can achieve better results. Another opportunity is the possibility of exploring new operators that could be used on the two-part chromosome. Due to the nature of the two-part chromosome, different pairings of operators used on the different parts of the chromosome may result in results superior to those obtained to date.

CONCLUSIONS

Once the computational advantages of the two-part chromosome were established, the method was used to solve a multi-line production scheduling problem that was modeled as an MTSP. Using the two-part method, an improvement over the existing scheduling method was observed. The two-part chromosome has been proven to be an effective representation of the MTSP. We have only scratched the surface of the possible implementations that may benefit from this method. Also an examination of improvements of the operators used on the two-part chromosome may provide additional improvements in the future.

Next we presented and modeled the traveling quality inspector problem. Using the custom GA, we produced significantly better results than were achieved with a commercially available GA package using two different chromosome representations. The custom GA provided a method of producing an effective schedule for the TQIP. Additional research into application of the custom GA as well as improvements to the custom GA are warranted.

References

Baita, F., Pesenti, R., Ukovich, W. and Favaretto, D. (2000). A comparison of different solution approaches to the vehicle scheduling problem in practical case, *Computers & Operations Research* 27(13), 1249-1269.

Bergey, P.K., Ragsdale, C.T. (1999). "Evolver 4.0: The Genetic Algorithm Super Solver," *OR/MS Today*, 44-47.

Bellmore, M. and Malone, J. (1971). Pathology of traveling-salesman subtour elimination algorithms, *Operations Research*, 19(2), 278-307.

Breedam, A. V. (2000), Comparing descent heuristics and metaheuristics for the vehicle routing problem, *Computers and Operation Research*, 28(4), 289-315.

Carter, A.E. and Ragsdale, C.T. (2002). Scheduling pre-printed newspaper advertising inserts using genetic algorithms, *Omega*, 30(6), 415-421.

Chatterjee, S., Carrera, C., Lynch, L. (1996). Genetic algorithms and traveling salesman problems. *European Journal of Operational Research.* 93(3), 490-510.

Davis, L., (1985). Job shop scheduling with genetic algorithms, *Proceedings of an International Conference on Genetic Algorithms*, London, 136-140.

Editor and Publisher (1998). Newspapers go postal over USPS's 'Auto Day,' *Editor and Publisher*, 131(25).

Evolver, Palisade Corporation, Newfield, NY (1998). (www.palisade.com).

E. W. Scripps (1999). 1999 Annual Report, Retrieved on March 23, 2001 from the World Wide Web: http://www.scripps.com/annrpt/99/nofrills/pdf/ew99.pdf.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, New York.

Fox, B.R., McMahon, M.B. 1991. Genetic operators for sequencing problems. *Foundations of Genetic Algorithms* (Edited by G. J. E. Rawlings). Morgan Kaugmann Publishers, San Mateo, California. pp. 284-300.

Garfinkel, R. and Nemhauser, G. (1972). *Integer Programming*, John Wiley & Sons, New York.

Glover, F. (1990). Artificial intelligence, heuristic frameworks and tabu search, *Managerial & Decision Economics*, 11(5), 365-378.

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

Goldberg, D.E., Lingle, R.J. (1985). Alleles, loci, and the traveling salesman problem, *Proceedings of An International Conference on Genetic Algorithms*, London, 154-159.

Goodrich, J. (2002). September 11, 2001 attack on America: a record of the immediate impacts and reactions in the USA travel and tourism industry, *Tourism Management* 23(6), 573-580.

Gouveia, L. and Pires, J. (2001). The asymmetric traveling salesman problem: on generalizations of disaggregated Miller-Tucker-Zemlin constraints, *Discrete Applied Mathematics*, 112(1-3), 129-124.

Holland, J.H. (1975). Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA.

Holland, J.H. (1992). Genetic Algorithms, *Scientific American*, 267(1), 66-72.

Jog, P., Suh, J.Y., Van Gucht, D. (1989). The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the traveling salesman problem, *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, 110-115.

Katayma, K and Sadamoto, H. (2000). The efficiency of hybrid mutation genetic algorithm for the traveling salesman problem, *Mathematical and Computer Modeling*, 31(10-12), 197-203.

Kimms, A. (1999). A genetic algorithm for multi-level, multi-machine lot sizing and scheduling, *Computers & Operations Research*, 26(8), 829-848.

Knosala, R. and Wal, T. (2001). A production scheduling problem using genetic algorithm, *Journal of Materials Processing Technology*, 109(1-2), 90-95.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A. and Shimoys, D. (1985). *The Traveling Salesman Problem*, John Wiley & Sons, New York.

Lenstra, J., Rinnooy Kay, A. (1975). Some simple applications of the traveling salesman problem, *Operational Research Quarterly*, 26(4), 717-733.

Liaw, C. (2000). A hybrid genetic algorithm for the open shop scheduling problem, *European Journal of Operational Research*, 124(1), 28-42.

Little, J., Murty, D., Sweeney, D. and Karel, C. (1963). An algorithm for the traveling salesman problem, *Operations Research*, 11(6), 972-989.

Malmborg, C. (1996). A genetic algorithm for service level based vehicle scheduling, *European Journal of Operational Research*, 93(1), 121-134.

Moon, C., Kim, J., Choi, G. and Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints, *European Journal of Operational Research*, 140(3), 606-617.

NAA (2001). Daily Newspaper Readership Trends, Retrieved on March 20, 2001 from the World Wide Web: http://www.naa.org/marketscope/databank/tdnpr.htm.

Neuwirth, R. (1998). Inserts Outpace ROP as Revenue Source, *Editor & Publisher*, 131(39), p. 8.

Ochi, L., Vianna, D., Drummond, L. and Victor, A. (1998). A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet, *Future Generation Computer Systems*, 14(5-6), 285-292.

Oliver, I., Smith, D. and Holland, J. (1987). A study of permutation crossover operators on the traveling salesman problem, *Proceedings of the second international conference on genetic algorithms*, London, 224-230.

Park, Y.B. (2001). A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines, *International Journal of Productions Economics*, 73(2), 175-188.

Poon, P.W., Carter, J.N. (1995). Genetic algorithm crossover operators for ordering applications, *Computers & Operations Research,* 22(1) 135-147.

Potvin, J. (1996). Genetic algorithms for the traveling salesman problems, *Annals of Operations Research*, 63, 330-370.

Premium Solver Platform Ver. 3.5, FrontLine Systems, Inc. (2001). Incline Village, NV, (www.frontsys.com).

Qu, L., Sun, R. 1999. A synergetic approach to genetic algorithms for solving traveling salesman problem, *Information Sciences,* 117(3-4) 267-283.

Ragsdale, C.T. (2001). *Spreadsheet Modeling and Decision Analysis*, 3rd edition, South-Western College Publishing, Cincinnati, Ohio.

Reeves, C.R. (1997). Genetic Algorithms for the Operations Researcher, *INFORMS Journal on Computing*, 9(3), 231-265.

Reinelt, G. (2001). TSPLIB, Retrieved Dec. 2002 from the World Wide Web: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

Renaud, J., Boctor, F. F., Ouenniche, J. (2000). A heuristic for the pickup and delivery traveling salesman problem, *Computers & Operations Research*, 27(3), 205-916.

Santos, A. and Dourado, A. (1999). Global optimization of energy and production in process industries: a genetic algorithm application, *Control Engineering Practice*, 7(4), 549-554.

Schaffer, J.D., Eshelman, L.J., Offutt, D. 1991. Spurious Correlations and Premature Convergence in Genetic Algorithms, *Foundations of Genetic Algorithms* (Edited by G. J. E. Rawlings). Morgan Kaugmann Publishers, San Mateo, California, 102-112.

Schmitt, L. and Amini, M. (1998). Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study, *European Journal of Operational Research*, 108(3), 551-570.

Shiromaru, I., Inuiguchi, M., Sakawa, M. (2000). A fuzzy satisficing method for electric power plant coal purchase using genetic algorithms, *European Journal of Operational Research*, 126(1), 218-230.

Shirrish, B., Nigel, J., Kabuka, M.R. (1993). A Boolean neural network approach for the traveling salesman problem, *IEEE Transactions on Computers* 42(10), 1271-1278.

Starkweather, T., Whitley, D., Whitley, C. & Mathial, K. (1991). A comparison of genetic sequencing operators, *Proceedings of the Fourth International Conference on Genetic Algorithms*, Los Altos, CA, 69-76.

Sweerda, G. (1991). *A Handbook of Genetic Algorithms*, Van Nostrand Reinhold, Amsterdam, 332-349.

Syswerda, G. (1989). Uniform Crossover in genetic algorithms, *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, 502-508.

Tang, L., Liu, J., Rong, A. and Yang, Z. (2000). A multiple traveling salesman problem model for hot rolling schedule in Shanghai Baoshan Iron & Steel Complex, *European Journal of Operational Research*, 124(2), 267-282.

Van Buer, M. G., Woodruff, D. L., Olson, R. T. (1999). Solving the medium newspaper production/distribution problem, *European Journal of Operational Research*, 115(2), 237-253.

Whitley, D., Starkweather, T., Fuquay, D. (1989). Scheduling problems and traveling salesmen: the genetic edge recombination operator, *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, 133-140.

Wang, D., Gen, M. and Cheng, R. (1999). Scheduling grouped jobs on single machine with genetic algorithm, *Computers & Industrial Engineering*, 26(2), 309-324.

Wong, R. (1980). Integer programming formulations of the traveling salesman problem, *Proceedings of the IEEE International Conference of Circuits and Computers*, 149-152.

APPENDIX A

Here, we offer mathematical proofs showing that the solution space of our new, two-part chromosome is smaller than those of the other two chromosomes summarized again below.

Technique	**Solution Space**
Two Chromosome:	$n!m^n$
One Chromosome:	$(n + m - 1)!$
Two-Part Chromosome:	$n!\binom{n-1}{m-1}$

We begin by showing (via Theorem 2.1) the solution space of the Two Chromosome technique exceeds that of the Two-Part Chromosome whenever n>m>1.

Lemma 2.1:

Given positive values A, B, C, and D, if A ≥ C and B ≥ D then AB ≥ CD.

Proof:

By contradiction, if AB < CD then B/D < C/A, but B/D ≥ 1 and C/A ≤ 1.

■ ■ ■

Theorem 2.1:

For integers $n > m \geq 1$, $n!m^n \geq n!\binom{n-1}{m-1}$.

Proof:

It suffices to show that $m^n \geq \binom{n-1}{m-1}$ for integers $n > m \geq 1$.

Let $n = m + k$ and assume $m^{m+k} \geq \binom{m+k-1}{m-1}$.

Observe that $m(k + 1) \geq (m + k)$ or $m \geq (m + k)/(k + 1)$ for all $m \geq 1$ and $k \geq 1$.

Thus, by Lemma 2.1,

$$mm^{m+k} \geq \frac{(m+k)}{(k+1)}\binom{m+k-1}{m-1} = \frac{(m+k)!}{(m-1)!(k+1)!} .$$

Hence,

$$m^{m+k+1} \geq \binom{m+k}{m-1}.$$

So, if Theorem 2.1 holds for any $n = m + k$, it holds for $n = m + k + 1$ and, by induction, any $n > m$.

Now, if $n = m + 1$, observe that $m^{m+1} \geq \binom{m}{m-1} = \frac{m!}{(m-1)!1!} = m$ for all $m \geq 1$.

Thus, Theorem (2.1) holds for all integers such that $n > m \geq 1$.

■ ■ ■

We now demonstrate (via Theorem 2.2) the solution space of the One Chromosome technique exceeds that of the Two-Part Chromosome whenever $n > m > 1$.

Lemma 2.2:

There are $\binom{n-1}{m-1}$ distinct *positive* integer-valued vectors $(x_1, x_2, \ldots, x_m)$ satisfying $x_1 + x_2 + \ldots + x_m = n$ where $x_i > 0$, $i = 1, 2, \ldots, m$. (Ross, 1984, page 14).

■ ■ ■

Lemma 2.3:

There are $\binom{n+m-1}{n}$ distinct *nonnegative* integer-valued vectors $(x_1, x_2, \ldots, x_m)$ satisfying $x_1 + x_2 + \ldots + x_m = n$ where $x_i \geq 0$, $i = 1, 2, \ldots, m$. (Ross, 1984, page 14).

■ ■ ■

Lemma 2.4:

For integers $n > m \geq 1$, $\binom{n+m-1}{n} \geq \binom{n-1}{m-1}$. (This follows immediately from Lemmas 2.2 and 2.3.)

■ ■ ■

Theorem 2.2:

For integers $n > m \geq 1$, $(n+m-1)! \geq n!\binom{n-1}{m-1}$.

Proof:

From Lemma 2.4, observe that $n!\binom{n+m-1}{n} \geq n!\binom{n-1}{m-1}$.

Thus, it suffices to show that $(n+m-1)! \geq n!\binom{n+m-1}{n}$.

By contradiction, suppose $(n+m-1)! < n!\binom{n+m-1}{n}$.

Then, $(n+m-1)! < n!\binom{n+m-1}{n} = \frac{(n+m-1)!}{(m-1)!}$,

which implies, $1 < \frac{1}{(m-1)!}$ for all $m \geq 1$.

However, $1 \geq \frac{1}{(m-1)!}$ for all $m \geq 1$.

Thus, $(n+m-1)! \geq n!\binom{n+m-1}{n} \geq n!\binom{n-1}{m-1}$.

■ ■ ■

Arthur E. Carter

Virginia Polytechnic Institute and State University
Pamplin College of Business
Department of Business Information Technology
web site: http://www.bit.vt.edu/Faculty/carter.htm
e-mail: arcarter@vt.edu

Education

- PhD in Business Information Technology, Virginia Polytechnic Institute and State University, Blacksburg VA, 2003.

- Masters of Business Administration, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2000.

- Bachelor of Science in Mechanical Engineering and Mechanics, Old Dominion University, Norfolk, VA, 1987.

Work Experience

- 2000 – Present, Graduate Assistant, Department of Business Information Technology, Virginia Tech, Blacksburg, VA.
- 1994 – 2000 Distribution Manager, The Roanoke Times, Roanoke, Virginia.
- 1988 – 1994 Nuclear Shift Test Engineer, Portsmouth, Virginia.

Teaching Experience

- BIT 2405: Quantitative Methods I (Va Tech)
- BIT 2406: Quantitative Methods II (Va Tech)
- Reactor Plant Fundamentals for Shift Test Engineers (Norfolk Naval Shipyard)

Doctoral Dissertation

Dissertation Title: Design and Application of Genetic Algorithms for the Multiple Traveling Salesperson Assignment Problem.

Published Refereed Journal Articles:

1. Carter, A.E. and Ragsdale, C.T. Scheduling Newspaper Advertising Inserts Using Genetic Algorithms, *OMEGA: The International Journal of Management Science*, 30(6), 415-421. .

Refereed Conference Proceedings:

1. Carter, A.E. and Ragsdale, C.T. (2002). Newspaper advertising insert scheduling using genetic algorithms, *Proceedings of the Thirty-second Annual Meeting of Southeast Region of Decision Sciences Institute*, 197-199, Hilton Head, SC.

2. Carter, A.E. (2001). Utilizing a genetic algorithm to improve the scheduling of pre-printed newspaper inserts, In the Proceedings of the Thirty-Seventh Annual Meeting of Southeastern INFORMS, Myrtle Beach, SC.

Other Articles:

1. Carter, A.E. and Ragsdale, C.T. (2001). A.I., in the Distribution Center, *TechNews*, Volume 7, Number 5, p. 28.

Awards and Honors:

1. 2002 Southeast Region Decision Sciences Institute Best Paper in the Production/Operations Management & TQM Track award winner.

2. 2001 Southeastern INFORMS, first place in the Ph.D. Student Paper Competition.

3. R. B. Pamplin Doctoral Fellowship.