



8-2012

# Solving Combinatorial Optimization Problems Using Genetic Algorithms and Ant Colony Optimization

Gautham Puttur Rajappa  
grajappa@utk.edu

---

## Recommended Citation

Rajappa, Gautham Puttur, "Solving Combinatorial Optimization Problems Using Genetic Algorithms and Ant Colony Optimization." PhD diss., University of Tennessee, 2012.  
[http://trace.tennessee.edu/utk\\_graddiss/1478](http://trace.tennessee.edu/utk_graddiss/1478)

This Dissertation is brought to you for free and open access by the Graduate School at Trace: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of Trace: Tennessee Research and Creative Exchange. For more information, please contact [trace@utk.edu](mailto:trace@utk.edu).

To the Graduate Council:

I am submitting herewith a dissertation written by Gautham Puttur Rajappa entitled "Solving Combinatorial Optimization Problems Using Genetic Algorithms and Ant Colony Optimization." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial Engineering.

Joseph H. Wilck, Major Professor

We have read this dissertation and recommend its acceptance:

Charles Noon, Xueping Li, Xiaoyan Zhu

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

---

# **Solving Combinatorial Optimization Problems Using Genetic Algorithms and Ant Colony Optimization**

**A Dissertation Presented for  
the Doctor of Philosophy  
Degree  
The University of Tennessee, Knoxville**

**Gautham Puttur Rajappa**

**August 2012**

Copyright © 2012 by Gautham P. Rajappa  
All rights reserved.

## **DEDICATION**

To my parents and friends

## ACKNOWLEDGEMENTS

First, I would like to express my gratitude to all my committee members.

- 1) **Dr. Joseph H. Wilck**, my advisor, for hiring me to pursue my Ph.D. at the University of Tennessee, Knoxville. You have been a source of inspiration to me and I whole heartedly thank you for supporting and challenging me for the past three years. I have had some of the most interesting conversations ranging from politics to sports with you.
- 2) **Dr. Charles Noon**, for taking his time out from his busy schedule and sharing his knowledge on my dissertation. I consider you as my role model.
- 3) **Dr. Xueping Li**, for providing your valuable inputs on my dissertation and also, for teaching some important courses which have helped me shape my career.
- 4) **Dr. Xiaoyan Zhu**, for providing your valuable inputs and pushing me to bring the best out of me.

Second, I would like to thank all the faculty members from the Departments of Industrial Engineering and Business Analytics. In particular, I would like to thank **Dr. Rapinder Sawhney**. You were there for me whenever I wanted to discuss anything personal or professional. You always answered me with a smile and some of your inputs have really helped me a lot in my personal life. Also, I would like to thank **Dr. John E. Bell** from the Business School for providing his valuable inputs on my dissertation. I would also like to thank you for helping me write my first ever journal paper. I honestly believe that the experience of sitting with you in your office and writing the paper gave me a whole new perspective of how journals paper have to be written.

Third, I would like to thank all my friends and colleagues, without whose support, I would never have been able to finish my Ph.D. My colleagues from UT are some the best students/friends I have ever worked with. In particular, I would like to thank my friends **Avik, Ajit, Aju, Karthik, Gagan, Sherin, Rani, Geetika, and Ashutosh** from Knoxville, who were always there for me and without whom, life would be very different in Knoxville. Also, I would like to thank my friends **Arjun & Sowmya** (for planning

some wonderful vacations), **Aubin** (my bank), **Priyamvad, Pai, Shailesh, Katti, Vincil, Ajay, Uday, Gidda, Sharath, Sarpa, Vinay** (for your motivating talks), **Durgesh** (for your perseverance), and **Ameya** (the smartest human being that I have ever known).

Finally, I would like to thank my parents **Shashikala and Rajappa, Bharath** (younger brother) and **Sandhya** (my older cousin sister), who always believed in me and supported me to pursue my dreams.

## ABSTRACT

This dissertation presents metaheuristic approaches in the areas of genetic algorithms and ant colony optimization to solve combinatorial optimization problems.

### ***Ant colony optimization for the split delivery vehicle routing problem***

An Ant Colony Optimization (ACO) based approach is presented to solve the Split Delivery Vehicle Routing Problem (SDVRP). SDVRP is a relaxation of the Capacitated Vehicle Routing Problem (CVRP) wherein a customer can be visited by more than one vehicle. The proposed ACO based algorithm is tested on benchmark problems previously published in the literature. The results indicate that the ACO based approach is competitive in both solution quality and solution time. In some instances, the ACO method achieves the best known results to date for the benchmark problems.

### ***Hybrid genetic algorithm for the split delivery vehicle routing problem (SDVRP)***

The Vehicle Routing Problem (VRP) is a combinatory optimization problem in the field of transportation and logistics. There are various variants of VRP which have been developed over the years; one of which is the Split Delivery Vehicle Routing Problem (SDVRP). The SDVRP allows customers to be assigned to multiple routes. A hybrid genetic algorithm comprising a combination of Ant Colony Optimization (ACO), Genetic Algorithm (GA), and heuristics is proposed and tested on benchmark SDVRP test problems.

### ***Genetic algorithm approach to solve the hospital physician scheduling problem***

Emergency departments have repeating 24-hour cycles of non-stationary Poisson arrivals and high levels of service time variation. The problem is to find a shift schedule that considers queuing effects and minimizes average patient waiting time and maximizes physicians' shift preference subject to constraints on shift start times, shift durations and total physician hours available per day. An approach that utilizes a genetic algorithm and discrete event simulation to solve the physician scheduling problem in a hospital is proposed. The approach is tested on real world datasets for physician schedules.

## TABLE OF CONTENTS

<b>CHAPTER I .....</b>	<b>1</b>
<b>Introduction.....</b>	<b>1</b>
1. Chapter Abstract .....	2
2. Metaheuristics Overview .....	2
3. Genetic Algorithms.....	3
3.1 Solving Multiobjective Optimization Problems with Genetic Algorithms.....	7
4. Ant Colony Optimization.....	10
4.1 ACO Algorithm .....	11
5. Dissertation Organization .....	13
6. References.....	14
<b>CHAPTER II.....</b>	<b>18</b>
<b>Ant Colony Optimization for the Split Delivery Vehicle Routing Problem .....</b>	<b>18</b>
Publication Statement .....	19
Chapter Abstract .....	19
1. Introduction.....	19
2. SDVRP Problem Formulation and Benchmark Data Sets .....	20
3. Ant Colony Optimization Approach.....	24
4. Computational experiments .....	30
5. Conclusions and Future directions.....	36
6. References.....	37
<b>CHAPTER III .....</b>	<b>40</b>
<b>A hybrid Genetic Algorithm approach to solve the Split Delivery vehicle routing problem .....</b>	<b>40</b>
Publication Statement .....	41
Chapter Abstract .....	41
1. Introduction.....	41
2. Split Delivery Vehicle Routing Problem (SDVRP) .....	42
3. Literature Review.....	44
4. Hybrid Genetic Algorithm Approach .....	46
4.1 Genetic Algorithms .....	46
5. Computation experiments .....	49
6. Conclusions and Future directions.....	54
7. References.....	55
<b>CHAPTER IV.....</b>	<b>57</b>
<b>A Genetic Algorithm approach to solve the physician scheduling problem.....</b>	<b>57</b>
Publication Statement .....	58
Abstract.....	58

1. Introduction.....	58
2. Literature Review.....	59
3. Problem Definition and Genetic Algorithm approach .....	63
3.1 Problem Definition.....	63
3.2 Genetic Algorithm Approach.....	69
4. Results, Conclusions, and Future Work.....	74
4.1 Results.....	74
4.2 Conclusions and Future Work .....	85
5. References.....	85
<b>CHAPTER V .....</b>	<b>90</b>
<b>Conclusion .....</b>	<b>90</b>
1. Chapter Abstract .....	91
2. Chapter Highlights .....	91
3. Future Directions .....	92
<b>VITA.....</b>	<b>93</b>

## LIST OF TABLES

Table 2.1: Parameters.....	31
Table 2.2: Comparing ACO results versus Jin et al. (2008) .....	31
Table 2.3: Comparing ACO results versus Chen et al. (2007a).....	32
Table 2.4: Post-hoc results (without using a candidate list) .....	34
Table 2.5: Comparison of ACO objective function for Chen et al. (2007a) and Column generation dual bound (Working paper, Wilck and Cavalier).....	35
Table 2.6: Comparison of ACO objective function for Jin et al. (2008) and Column generation dual bound (Working paper, Wilck and Cavalier).....	36
Table 3.1: Parameters.....	51
Table 3.2: Comparing Hybrid GA results versus Jin et al.(2008d) .....	51
Table 3.3: Comparing Hybrid GA results versus Chen et al. (2007c).....	52
Table 3.4: Comparison of ACO objective function for Chen et al. (2007c) and Column generation dual bound (Working paper, Wilck and Cavalier).....	53
Table 3.5: Comparison of ACO objective function for Jin et al. (2008d) and Column generation dual bound (Working paper, Wilck and Cavalier).....	54
Table 4.1: Given Data .....	64
Table 4.2: Average number of patients arriving per hour (Dataset 1) .....	65
Table 4.3: Average number of patients arriving per hour (Dataset 2) .....	65
Table 4.4: Feasible shifts with preference (Dataset 1).....	66
Table 4.5: Feasible shifts with preference (Dataset 2).....	66
Table 4.6: Shift index (Shift preference) matrix (Dataset 1) .....	70
Table 4.7: Shift index (Shift preference) matrix (Dataset 2) .....	74
Table 4.8: Weighted sum approach results (Dataset 1) .....	75
Table 4.9: Weighted sum approach results (Dataset 2) .....	76
Table 4.10: Number of patients of capacity (Dataset 1) .....	80
Table 4.11: Number of patients of capacity (Dataset 2) .....	83

## LIST OF FIGURES

Figure 1.1: Genetic Algorithm Flowchart.....	4
Figure 1.2: Ant Colony Optimization .....	11
Figure 2.1: ACO Flowchart .....	28
Figure 3.1: CVRP v/s SDVRP .....	43
Figure 3.2: Hybrid GA Flowchart.....	50
Figure 4.1: Genetic Algorithm Flowchart (Dataset 1) .....	73
Figure 4.2: Total preference violation v/s Average patient wait time (min)(Dataset 1)...	78
Figure 4.3: Total preference violation v/s Average patient wait time (min)(Dataset 2)...	78
Figure 4.4(A): Number of patients of capacity plot (Case # 2, Dataset 1) .....	81
Figure 4.4(B): Number of patients of capacity plot (Case # 6, Dataset 1) .....	81
Figure 4.4(C): Number of patients of capacity plot (Case # 11, Dataset 1) .....	82
Figure 4.5(A): Number of patients of capacity plot (Case # 2, Dataset 2) .....	84
Figure 4.5(B): Number of patients of capacity plot (Case # 6, Dataset 2) .....	84
Figure 4.5(C): Number of patients of capacity plot (Case # 11, Dataset 2) .....	85

**CHAPTER I**  
**INTRODUCTION**

## 1. Chapter Abstract

In this chapter, a brief overview on metaheuristics is presented. Since, this dissertation focuses on Genetic Algorithms and Ant Colony Optimization, a detailed overview of both the metaheuristics is provided in the chapter.

## 2. Metaheuristics Overview

A large number of well-known numerical combinatorial programming, linear programming (LP), and nonlinear programming (NLP) based algorithms are applied to solve a variety of optimization problems. In small and simple models, these algorithms were always successful in determining the global optimum. But in reality, many optimization problems are complex and complicated to solve using algorithms based on LP and NLP methods. Combinatorial optimization (Osman and Kelly, 1996a) can be defined as a mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in number. A combinatory optimization problem can be either easy or hard. We call the problem *easy* if we can develop an efficient algorithm to solve for optimality in a polynomial time. If an efficient algorithm does not exist to solve for optimality in a polynomial time, we call the problem *hard*. An optimal algorithm to compute optimality for *hard* problems requires a large number of computational steps which grows exponentially with the problem size. The computational drawbacks of such algorithms for complex problems have led researchers to develop metaheuristic algorithms to obtain a (near) optimal solution.

The term "metaheuristic" was first coined by Fred Glover (1986). Generally, it is applied to problems classified as NP-Hard or NP-Complete but could also be applied to other combinatorial optimization problems. Metaheuristics are among the best known methods for a good enough and cheap (i.e., minimal computer time) solution for NP-Hard or NP-Complete problems. Some of the typical examples where metaheuristics are used are the traveling salesman problem (TSP), scheduling problems, assignment problems, and vehicle routing problems (VRP). Such types of problems falls under combinatory optimization problems. According to Osman and Laporte (1996b), a metaheuristic

algorithm is defined as: "An iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions." According to Blum and Roli (2003a), metaheuristics are strategies that guide a search process which explore the search space to find a (near-) optimal solution. Metaheuristics are not problem-specific and may make use of domain-specific knowledge in the form of heuristics. Some of the well known metaheuristic approaches are genetic algorithm, simulated annealing, Tabu search, memetic algorithm, ant colony optimization, particle swarm optimization, etc. The following sections provide an overview of Genetic Algorithms and Ant Colony Optimization, which are relevant to this dissertation.

### 3. Genetic Algorithms

Genetic algorithms are population based search algorithms to solve combinatorial optimization problems. It was first proposed by John Holland (1989). They generate solutions for optimization problem based on theory of evolution using concepts such as reproduction, crossover and mutation. The fundamental concept of a genetic algorithm states a set of conditions to achieve global optima. These conditions describe the reproduction process and ensure that better solution remain in future generations and weaker solutions be eliminated from future generations. This is similar to the Darwin's survival of fittest concept in the theory of evolution. A typical genetic algorithm (GA) consists of the following steps (Holland, 1989):

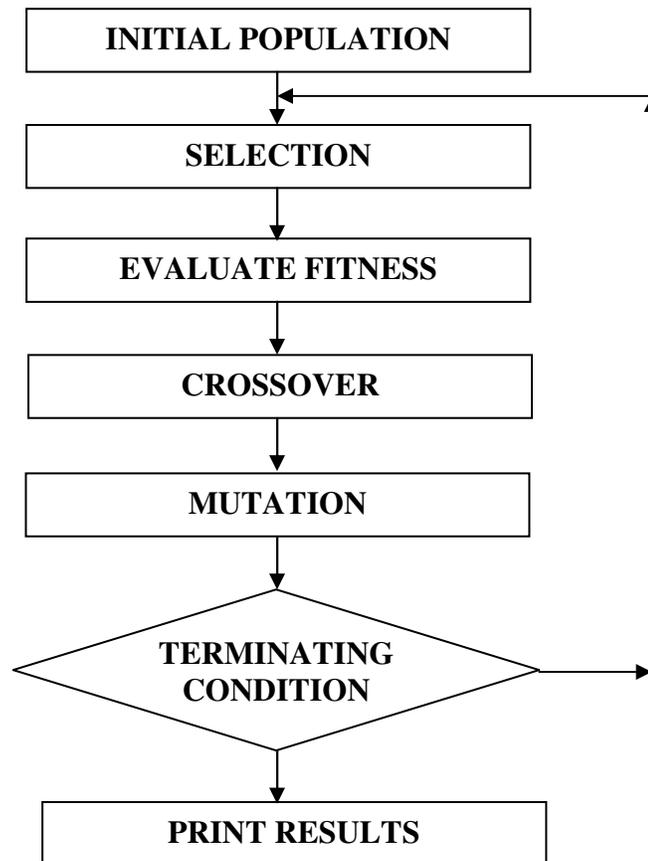
- Step 1:** Generate an initial population of  $N$  solutions.
- Step 2:** Evaluate each solution of the initial population using a fitness function/objective function.
- Step 3:** Select solutions as parents for the new generation based on probability or randomness. The best solutions (in terms of fitness or objective) have a higher probability of being selected than poor solutions.

**Step 4:** Use the parent solutions from Step 3 to produce the next generation (called offspring). This process is called as crossover. The offspring are placed in the initial set of solutions replacing the weaker solutions.

**Step 5:** Randomly alter the new generation by mutation. Usually this is done using a mutation probability.

**Step 6:** Repeat Steps 2 through 5 until a stopping criteria is met.

A flowchart of a simple GA is shown in Figure 1.1 below:



**Figure 1.1: Genetic Algorithm Flowchart**

A genetic algorithm search mechanism consists of three phases: (1) Evaluation of fitness function of each solution in the population, (2) selection of parent solutions based on fitness values, and (3) application of genetic operations such as crossover and mutation to generate new offspring.

The initial population in genetic algorithm is normally generated randomly but heuristic approaches can also be applied to get a good set of initial solutions for the initial population. Genetic operations involve crossover and mutation. In a crossover operation, one or two points in the parent string are cut at random and the properties are exchanged between two parents to generate two or four offspring. For example, consider two binary parents represented by Parent 1: 1-0-0-1 and Parent 2: 1-1-0-0. A crossover can occur at any point(s) between each element of the parent. Based on probability (i.e., generating a random number between 0 and 1), a crossover point is chosen. For example, if the crossover point was after the second position for the above parents. Then, the two new offspring are generated as follows: Offspring 1: 1-0-0-0 and Offspring 2: 1-1-0-1. These offspring inherit certain characteristics from their parents.

There are various crossover techniques that are described in literature such as one-point crossover, two-point crossover, multi point crossover, variable to variable crossover and uniform crossover (HasancEebi and Erbatur, 2000). In one-point crossover, a single point is selected in the parent string and crossover operation is performed. In two-point crossover, two points are selected in the parent string and crossover is performed accordingly. In multi point crossover, more than two points are selected randomly and crossover is performed. In variable to variable crossover, the parents are divided into substrings and a one point crossover is performed for each substring. In uniform crossover, randomly generated crossover masks are first created. Then for the child, wherever there is one in the mask, the genes are copied from parent 1 and for zeros, the genes are copied from parent 2. The second child is created either by complementing the original mask or by creating a new crossover mask.

Once the crossover operations performed, mutation is done to prevent the genetic algorithm from being trapped in local optima (Osman and Kelly, 1996a). But the mutation probability is kept low to avoid delay in convergence to global optima. In the mutation stage, again using the concept of probability, an offspring will be selected and all or some of its positional values will be changed. For example, consider applying

mutation on Offspring 1: 1-0-0-0. After applying mutation, the new Offspring 1: 0-1-1-1 will be formed. There is also a concept called elitism in genetic algorithm. If elitism is used, the fittest parent(s) are directly copied to the new population.

Problems for generating feasible offspring are problem specific and hence, the application of crossover and mutation operators also differs. Also, due to constraints of a particular problem, pure genetic algorithms cannot be applied to obtain a feasible set of solutions. In such cases, to ensure feasibility, additional procedures are used to ensure feasibility based on the specific problem's constraints.

Over a period of time, a lot of variants of genetic algorithms have been developed. Adaptive Genetic Algorithms (AGA) (Srinivas & Patnaik, 1994) is one of the most significant variant of genetic algorithm. In a normal GA, the crossover and mutation probabilities are fixed. The selection of this probability is significant because it decides on the convergence rate and the accuracy of the solution. Usually crossover probabilities are fixed between 0.6 and 0.8 and the mutation probability is between 1-3%. An AGA in turn dynamically changes the crossover and mutation probability based on the fitness value of the new generation. This real time manipulation of these probabilities aids in better convergence and maintaining a diverse population. Some of the recent application of adaptive genetic algorithm are bilateral multi-issue simultaneous bidding negotiation (2008) and designing and optimizing phase plates for shaping partially coherent beams (March 2010). Another variant is the multiobjective genetic algorithm, which is explained in the section 3.1.

Some of the most recent applications of genetic algorithms are in deployment of security guards in a company (Dec 2010a), optimizing the design of spur gears (2010c), electric voltage stability assessment (2010a), capacitated plant location problem (2010b), evaluation of RFID applications (Nov 2010b), supply chain management to coordinate production and distribution (Dec 2010b), and forecasting of energy consumption (Nov 2010a).

### ***3.1 Solving Multiobjective Optimization Problems with Genetic Algorithms***

In the real world, there are an infinite number of problems that require more than one objective to be simultaneously satisfied under a given set of constraints. Such problems fall under the category of multiobjective optimization problems. Multiobjective optimization problems can be found in various fields: oil and gas industries, finance, aircraft, and automobile design.

Consider a minimization problem consisting of  $N$  objectives with a series of constraints and bounds on decision variables. Given an  $n$  dimensional decision variables vector, the goal is to find a vector in solution space that minimizes the given set of  $N$  objective function (2002a, 2006). Examples of the objectives to be simultaneously solved would be maximizing profit while minimizing costs, maximizing the fuel efficiency but not compromising on performance. In certain cases, objective functions may be optimized independently, but generally objectives must be simultaneously optimized to reach a reasonable solution that compromises the multiple objectives. Instead of a single solution that simultaneously minimizes each objective function, the aim of a multiobjective problem is to determine a set of non-dominated solutions, known as Pareto-optimal (PO) solutions (2002a). A Pareto optimal set is a set of solutions that are non-dominated with respect to each other. While traversing from one solution to another in a Pareto set, there is always a certain amount of compromise in one objective(s) with respect to improvement in other objective(s). Finding a set of such solutions and then comparing them with one another is the primary goal of solving multiobjective optimization problems.

In the real world, it is impossible to optimize all the objective functions simultaneously. A traditional multiobjective optimization approach aggregates together (e.g., by normalizing, using weights) various objectives to form a single overall fitness function, which can then be treated by classical techniques such as simple GAs, multiple objective linear programming (MOLP), random search, etc. But using such aggregate approaches produces results which are sensitive to the weights selected. Hence, the goal of a

multiobjective optimization problem is the find a set of solutions, each of which satisfies all the objective functions at an acceptable level and are non-dominated by other solutions. These set of solutions are called Pareto optimal set and the corresponding objective function values are called Pareto front (1985a). The size of the Pareto optimal set depends on the size of a problem and hence, it is difficult to find the entire Pareto-Optimal set for larger problems. Also, in combinatory optimization problems, it generally impossible to compute the evidence of a Pareto optimal set.

There are numerous approaches provided in the literature to solve multiobjective optimization problems. One approach is to combine the individual objective functions into a single composite function by weighting the objectives with a weight vector (2006). The results obtained from this approach largely depend on the weights selected and proper selecting of weights can has a major impact on the final solution. The primary drawback of this approach is that instead of returning a set of solutions, it returns a single solution. Another approach is to determine an entire Pareto optimal solution set, or a representative subset, and is a preferred approach to solve real world multiobjective optimization solutions (2006). Some of the most well known operations research approaches to solve multiobjective problems are efficient frontier, goal programming, game theory, Gradient Based/Hill Climbing, Q-Analysis, and compromise programming (2002b).

Conventional optimization techniques such as simplex-based methods and simulated annealing are not designed to solve problems with multiple objectives. In such cases, multiobjective problems have to be reformulated as a single-objective optimization problem which results in a single solution per run of the optimization solver. However, evolutionary algorithms (EAs) such as genetic algorithms can be applied to solve such problems. Genetic algorithms are population based search algorithms and can be used to solve multiobjective optimization problems. Genetic Algorithms can solve such problems by using specialized fitness functions and introducing methods to promote solution diversity (2006).

When applying genetic algorithms (GA) to a problem with a single objective function, we randomly select a set of individuals (chromosomes) to form the initial population. We then evaluate their fitness functions. Using this initial population, we then create a new population by incorporating mutation and crossover operations and then, repeat the process of fitness evaluation and crossover-mutation process over many generations with a hope of converging to the global optimum. In traditional single-objective GA approach to solve multiobjective problems, we can combine the individual objective functions into a single composite function by weighting the objectives with a weight vector. Another approach is to make most of the objectives as varying constraints and optimize just the main objective. Both these approaches require multiple runs to generate Pareto-optimal solutions consecutively. But the ability of GA to simultaneously search different regions of a solution space makes it possible for a generic single-objective GA to be modified into a multiobjective GA to find a set of Pareto optimal solutions in one run. In addition, most multiobjective GAs do not require the user to prioritize, scale, or weight objectives. Therefore, GAs is one of the most frequently used metaheuristics to solve multiobjective optimization problems. In fact, 70% of the metaheuristics approaches used to solve multiobjective optimization problems uses genetic algorithms (2002b).

The fundamental goals in multiobjective genetic algorithm design are:

- Directing the search towards the Pareto set (fitness assignment and selection),
- Maintaining a diverse set of Pareto solutions( diversity), and
- Retaining the best chromosomes in future generations (elitism) (2004b) with computational speed being another important criterion.

Some of the well known variants of multiobjective genetic algorithms are listed below:

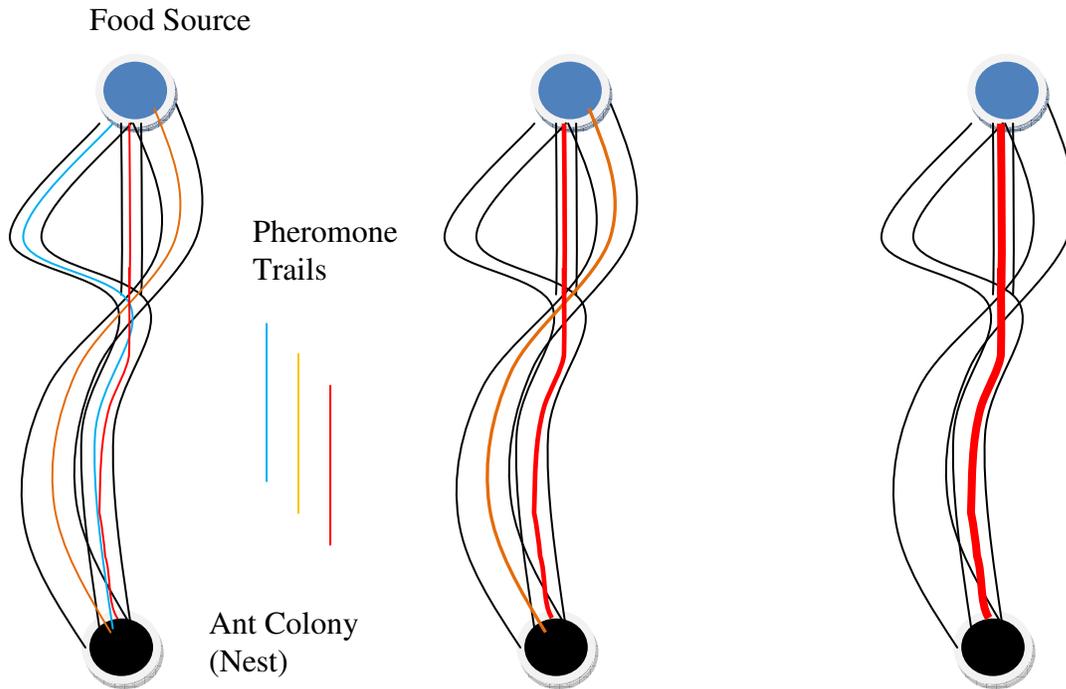
- The first multiobjective genetic algorithm called vector evaluated genetic algorithm (VEGA) was developed by Schaffer (1985b). It mainly focused on the fitness selection and did not address the issues related with maintaining diversity and elitism.

- Multiobjective Genetic Algorithm (MOGA) (1993a; 1993b) used Pareto ranking and fitness sharing by niching for fitness selection and maintenance of diversity respectively.
- Hajela & Lin's Weighting-based Genetic Algorithm (HLGA) (1992b) is based on assigning weights to each normalized objectives.
- Non-dominated Sorting Genetic Algorithm (NSGA) (1995) in which the fitness assignment was based on Pareto fitness sharing and diversity was maintained by niching.
- Niche Pareto Genetic Algorithm (NPGA) (June 1994) in which diversity is based on tournament selection criteria.
- Pareto-Archived Evolution Strategy (PAES) (1999b) in which Pareto dominance rule is used to replace a parent in the new population.

#### **4. Ant Colony Optimization**

Ant Colony Optimization (ACO) is a metaheuristic approach proposed by Dorigo (1992a) in 1992 to solve combinatorial optimization problems. Inspired by the behavior of ants forming pheromone (e.g., a trace of a chemical substance that can be smelled by other ants (Rizzoli et al. , 2004a)) trails in search of food, ACO belongs to a class of algorithms which can be used to obtain good enough solutions in reasonable computational time for combinatorial optimization problems. Ants communicate with one another by depositing pheromones. Initially in search of food, ants wander randomly and upon finding a food source, return to their colony. On their way back to the colony, they deposit pheromones on the trail. Other ants then tend to follow this pheromone trail to the food source and on their way back may either take a new trail, which might be shorter or longer than the previous trail, or would come back along the previous laid pheromone trail. Also, on their way back, the other ants deposit pheromones on the trail. Pheromones have a tendency to evaporate with time. Hence, over a period of time, the shortest trail (path) from the food source to the colony would become more attractive and have a larger amount of pheromone deposited as compared with other trails. A pictorial explaining of the above defined steps is shown in Figure 1.2 below. Initially, a single ant,

called "blitz," goes from the colony to the food source via the blue pheromone trail. As time progresses, more and more ants either follow this blue trail or form their own shorter trail (red and orange trail). Eventually, the shortest trail (red) becomes more attractive and is taken by all the ants from the colony to the food source and the other trails evaporate in a period of time (2004a).



**Figure 1.2: Ant Colony Optimization**

#### **4.1 ACO Algorithm**

The ACO replicates the foraging behavior of ants to construct a solution. The main elements in an ACO are ants which independently build solutions to the problem. For an ant  $k$ , the probability of it visiting a node  $j$  after visiting node  $i$ , depends on the two attributes namely:

- **Attractiveness ( $\mu_{ij}$ ):** It is a static heuristic value that never changes. In the case of VRP, it is calculated as inverse of arc length for shortest path problems and for other variants, it can depend on other parameters besides the arc length (e.g., in

VRPTW it also depends on the current time and the time window limits of the customers to be visited (2004a).

- **Pheromone trails( $\tau_{ij}$ ):** It is the dynamic component which changes with time. It is used to measure the desirability of insertion of an arc in the solution. In other words, if an ant finds a strong pheromone trail leading to a particular node, that direction will be more desirable than other directions. The trail desirability depends on the amount of pheromone deposited on a particular arc (2004a).

The probability of an unvisited node  $j$  being selected after node  $i$  is according to a random-proportional rule (2004a):

$$P_{ij}^k(t) = \frac{[(\tau_{ij}(t))^\alpha][(\mu_{ij}^\beta)]}{\sum_{i \in N_i^k} [(\tau_{ij}(t))^\alpha][(\mu_{ij}^\beta)]} \quad \text{if } j \in N_i^k \quad (1.1)$$

Where  $\mu_{ij} = 1/d_{ij}$ , where  $d_{ij}$  is the length of arc,  $\alpha$  and  $\beta$  are which determine the relative influence of pheromone trail and heuristic information respectively,  $N_i^k$  is the feasible neighborhood of  $k$  (i.e., nodes not yet visited by  $k$ ).

The pheromone information on a particular arc ( $i,j$ ) is updated in the pheromone matrix using the following equation:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij} + \sum_{i=1}^m \Delta \tau_{ij}^k(t) \quad (1.2)$$

Where  $0 \leq \rho \leq 1$  the pheromone trail evaporation rate and  $m$  is the number of ants. Trail evaporation also occurs after each iteration, usually by exponential decay to avoid locking into local minima (2004a).

After each iteration, the best solution found is used to update the pheromone trail. This procedure is repeated again and again until a terminating condition is met. In ACO, the pheromone trail is updated locally during solution construction and globally at the end of construction phase. An interesting aspect of pheromone trail updating is that every time

an arc is visited, its value is diminished which favors the exploration of other non visited nodes and diversity in the solution (2004a).

There is an another optional component called Daemon actions which are used to perform centralized actions such as calling a local search procedure or collect global information to deposit addition pheromones on edges from a non-local perspective. Pheromone updates performed by daemons are called off-line pheromone updates (2004a).

The ACO pseudo-code for ACO is described below:

```
Procedure ACO  
    While (terminating condition is not met)  
        Generate_solutions()  
        Pheromone_Update()  
        Daemon_Actions() // this is optional  
    End while  
End procedure
```

Some of the more recent application where ACO is applied are in multimode resource-constrained project scheduling problem (MRCPSP) with the objective of minimizing project duration (Zhang, 2012a), inducing decision trees (Otero et al., 2012b), wherein traditional ACO algorithm is developed combining the traditional decision tree induction algorithm and ACO, and Robot path planning (Bai et al., 2012c).

## **5. Dissertation Organization**

The rest of the dissertation is organized as follows. Chapter II discusses literature, an ant colony optimization procedure, and computational results for the split delivery vehicle routing problem. Chapter III discusses literature, a hybrid genetic algorithm procedure, and computational results for the split delivery vehicle routing problem. Chapter IV discusses literature and a genetic algorithm approach to solve a specific hospital physician scheduling problem. Summary and future works are presented in Chapter V.

Also, references for each chapter of the dissertation are provided at the end of each chapter.

## 6. References

- Bai, J., Chen, L., Jin, H., Chen, R., & Haitao Mao, H. (2012c) , Robot Path Planning Based on Random Expansion of Ant Colony Optimization, *Lecture Notes in Electrical Engineering, Recent Advances in Computer Science and Information Engineering*, 125, 141-146.
- Blum, C., & Andrea, R. (2003a). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR)*, 35(3), 268–308.
- Delavar, M.R., Hajiaghaei-Keshteli, M., & Molla-Alizadeh-Zavardehi, S. (Dec 2010b). Genetic algorithms for coordinated scheduling of production and air transportation. *Expert Systems With Applications*, 37(12), 8255-8266.
- Devaraj, D., & Roselyn, J. P. (2010a). Genetic algorithm based reactive power dispatch for voltage stability improvement. *International Journal of Electrical Power and Energy Systems*, 32(10), 1151-1156.
- Dias, A.H.F., & Vasconcelos, J.A.de. (2002a). Multiobjective genetic algorithms applied to solve optimization problems. *IEEE Transactions on Magnetics*, 38(2), 1133-1136.
- Dorigo, M. (1992a). *Ph.D. Thesis, Optimization, learning and natural algorithms (in Italian)*. Politecnico di Milano, Italy.
- Otero, F.E.B., Freitas, A.A., & Johnson, C.G. (2012b), Inducing decision trees with an ant colony optimization algorithm, *Applied Soft Computing*.
- Fonseca, C.M., & Fleming, P.J. (1993a). *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*. Paper presented at the Fifth International Conference on Genetic Algorithms, San Francisco, CA, USA, 416-423.

- Fonseca, C.M & Fleming, P.J. (1993b). *Multiobjective genetic algorithms*. Paper presented at the IEE colloquium on ‘Genetic Algorithms for Control Systems Engineering’, 193(130), London, UK.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5), 533–549.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.)
- Hajela, P. & Lin, C.-Y., (1992b). Genetic search strategies in multicriterion optimal design *Structural and Multidisciplinary Optimization*, 4(2), 99-107.
- Hasanc ebi, O., & Erbatur, F. (2000) , Evaluation of crossover techniques in genetic algorithm based optimum structural design, *Computers and Structures*, 78, 435-448.
- Horn, J. Nafpliotis, N., & Goldberg, D.E., (June 1994). *A niched Pareto genetic algorithm for multiobjective optimization*. Evolutionary Computation, Paper presented at the IEEE world congress on Computational Intelligence, Orlando, FL, USA, 82-87.
- Jian, L., Wang, C., & Yi-xian, Y. (2008). An adaptive genetic algorithm and its application in bilateral multi-issue negotiation. *The Journal of China Universities of Posts and Telecommunications*, 15, 94-97.
- Jones, D.F., Mirrazavi, S.K., & Tamiz, M. (2002b). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1), 1-9.
- Knowles, J. & Corne, D. (1999b). *The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation*. *IEEE Press, Paper presented at the Congress on Evolutionary Computation (CEC99)*, 1, Piscataway, NJ, 98–105.
- Konak, A., Coit, D.W., & Smith, A.E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91(9), 992–1007.

- Lai, M-C., Sohn, H-S., Tseng, T-L., & Chiang, C. (2010b). A hybrid algorithm for capacitated plant location problem. *Expert Systems With Applications*, 37(12), 8599-8605.
- Lau, H.C.W, Ho, G.T.S., Zhao, Y., & Hon, W.T. (Dec 2010a). Optimizing patrol force deployment using a genetic algorithm. *Expert Systems With Applications*, 37(12), 8148-8154.
- Li, J., Zhu, S., & Lu, B. (March 2010). Design and optimization of phase plates for shaping partially coherent beams by adaptive genetic algorithms. *Optics and Laser Technology*, 42(2), 317-321.
- Li, K., & Su, H. (Nov 2010a). Forecasting building energy consumption with hybrid genetic algorithm-hierarchical adaptive network-based fuzzy inference system. *Energy & Buildings*, 42(11), 2070-2076.
- Mendi, F., Baskal, T., Boran, K., & Boran, F. E. (2010c). Optimization of module, shaft diameter and rolling bearing for spur gear through genetic algorithm. *Expert Systems With Applications*, 37(12), 8058-8064.
- Osman, I.H, & Kelly, J.P. (1996a). *Meta-heuristics. Theory and Applications* (Kluwer, Boston).
- Osman, I. H., & Laporte, G. (1996b). "Metaheuristics: A bibliography". *Annals of Operations Res*, 63(5), 513–623.
- Osyczka, A. (1985a). Multicriteria *optimization for engineering design*," J. S. Gero, Ed. Academic Press, Inc., New York, NY, 193–227.
- Rizzoli, A. E., Oliverio, F., Montemanni, R., & Gambardella, L. M. (2004a). Ant colony optimisation for vehicle routing problem: from theory to applications. *Technical Report TR-15-04*.
- Schaffer, J.D. (1985b). *Multiple objective optimization with vector evaluated genetic algorithms*. Paper presented at the Proceedings of 1st International Conference on Genetic Algorithms , Pittsburgh, PA, 93-100.
- Srinivas, M. & Patnaik, L.M. (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4), 656-667.

- Srinivas, N. and Deb, K. (1995) Multi-Objective function optimization using non-dominated sorting genetic algorithms, *Evolutionary Computation*, 2(3),221–248.
- Trappey, A.J.C., Trappey, C.V., & Wu, C.-R. (Nov 2010b). Genetic algorithm dynamic performance evaluation for RFID reverse logistic management. *Expert Systems With Applications*, 37(11), 7329-7335.
- Zhang, H. (2012a). "Ant Colony Optimization for Multimode Resource-Constrained Project Scheduling." *J. Manage. Eng.*, 28(2), 150–159.
- Zitzler, E., Laumanns, M., & Bleuler, S. (2004b). *In X. Gandibleux and others, editors, A Tutorial on Evolutionary Multiobjective Optimization*, Lecture Notes in Economics and Mathematical Systems, Springer.

**CHAPTER II**  
**ANT COLONY OPTIMIZATION FOR THE SPLIT DELIVERY**  
**VEHICLE ROUTING PROBLEM**

## **Publication Statement**

This paper is a joint work between Gautham P. Rajappa, Dr. Joseph H. Wilck, and Dr. John E. Bell. Currently, we are working on the paper for publication. To the best of our knowledge, ACO has never been applied to SDVRP and hence, we intend to publish this paper in near future.

## **Chapter Abstract**

An Ant Colony Optimization (ACO) based approach is presented to solve the Split Delivery Vehicle Routing Problem (SDVRP). SDVRP is a relaxation of the Capacitated Vehicle Routing Problem (CVRP) wherein a customer can be visited by more than one vehicle. The proposed ACO based algorithm is tested on benchmark problems previously published in the literature. The results indicate that the ACO based approach is competitive in both solution quality and solution time. In some instances, the ACO method achieves the best known results to date for some benchmark problems.

## **1. Introduction**

The Vehicle Routing Problem (VRP) is a prominent problem in the fields of logistics and transportation. With an objective to minimize the delivery cost of goods to a set of customers from depot(s), numerous variants of the VRP have been developed and studied over the years. One such variant is the Split Delivery Vehicle Routing Problem (SDVRP) which is a relaxation of the Capacitated Vehicle Routing Problem (CVRP). In the case of a CVRP, each customer is served by only one vehicle, whereas in SDVRP, the customer demand can be split between vehicles. For example, consider three customers each with a demand of 100 served by vehicle with a capacity of 150. In the case of the CVRP, three vehicles are required but in the case of SDVRP, since the customer demand can be split amongst multiple vehicles, only two vehicles are required to fulfill the customer demand. SDVRP was first developed by Dror and Trudeau (1989; 1990). They showed that if the demand is relatively low compared to the vehicle capacity and the triangular inequality holds, an optimal solution exists in the SDVRP in which two routes cannot have more than one common customer. In addition, it was proven that the

SDVRP is NP-hard and has potential in savings in terms of the distance traveled as well as the number of vehicles used.

Over the past few years, several metaheuristics such as Genetic Algorithms and Tabu Search were applied to solve SDVRP. However, to the best of my knowledge, no journal article has applied and experimentally tested the ability of the ACO algorithm on SDVRP instances. Hence, I developed an ACO for SDVRP and test the capability of my algorithm on benchmark test problems.

The rest of the chapter is organized as follows: Section 2 and Section 3 provide an overview of SDVRP and ACO algorithm respectively. Computational experiments are described in Section 4. Conclusions and future work are summarized in Section 5.

## **2. SDVRP Problem Formulation and Benchmark Data Sets**

In this section, I present the problem formulation and discuss the relevant literature for SDVRP.

According to Aleman et al. (2010b), the SDVRP is defined on an undirected graph  $G = (V, E)$  where  $V$  is the set of  $n + 1$  nodes of the graph and  $E = \{(i, j) : i, j \in V, i < j\}$  is the set of edges connecting the nodes. Node 0 represents a depot where a fleet  $M$  of identical vehicles with capacity  $Q$  are stationed, while the remaining node set  $N = \{1, \dots, n\}$  represents the customers. A non-negative cost, usually a function of distance or travel time,  $c_{ij}$  is associated with every edge  $(i, j)$ . Each customer  $i \in N$  has a demand of  $q_i$  units. The optimization problem is to determine which customers are served by each vehicle and what route the vehicle will follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as travel distance, gas consumption, and vehicle depreciation. The most frequently used formulations for SDVRP found in literature are from Dror and Tredeau (1990), Frizzell and Giffin (1992b), and Dror et al. (1994).

I use the SDVRP flow formulation from Wilck and Rajappa (2010c) which is given below. This formulation assumes that  $c_{ij}$  satisfies the triangle inequality and that exactly the minimum number of vehicle routes,  $K$ , are used. The formulation does not assume that distances are symmetric.

**Indexed Sets:**

$i = \{1, 2, \dots, n\}$  ; node index ; 1 is the depot

$j = \{1, 2, \dots, n\}$  ; node index

$k = \{1, 2, \dots, m\}$  ; route index

**Parameters:**

$m$  : The number of vehicle routes

$n$  : The number of nodes

$Q$  : The vehicle capacity

$c_{ij}$  : The cost or distance from node  $i$  to node  $j$

$d_i$  : The demand of customer  $i$ , where  $d_1 = 0$ .

**Decision Variables:**

$x_{ijk}$  : A binary variable that is one when arc  $(i, j)$  is traversed on route  $k$ ; zero otherwise

$u_{ik}$  : Free variable used in the sub-tour elimination constraints

$y_{ik}$  : A binary variable that is one when node  $i$  is visited on route  $k$ ; zero otherwise

$v_{ik}$  : A variable that denotes the amount of material delivered to node  $i$  on route  $k$

Without loss of generality,  $y_{ik}$  and  $v_{ik}$  are not defined for  $i = 1$ .

**Objective: Minimize Travel Distance**

$$\text{Minimize } Z = \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n c_{ij} \sum_{k=1}^m x_{ijk} \tag{2.1}$$

### Constraints:

$$\sum_{k=1}^m v_{ik} = d_i, \forall i = 2, \dots, n \quad (2.2)$$

$$\sum_{i=2}^n v_{ik} \leq Q, \forall k = 1, \dots, m \quad (2.3)$$

$$\sum_{\substack{i=1 \\ i \neq p}}^n x_{ipk} - \sum_{\substack{j=1 \\ j \neq p}}^n x_{pjk} = 0, \forall k = 1, \dots, m; p = 1, \dots, n \quad (2.4)$$

$$u_{ik} - u_{jk} + nx_{ijk} \leq n - 1, \forall i = 2, \dots, n; i \neq j; k = 1, \dots, m \quad (2.5)$$

$$d_i y_{ik} \geq v_{ik}, \forall k = 1, \dots, m; i = 2, \dots, n \quad (2.6)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^n x_{ijk} = y_{ik}, \forall k = 1, \dots, m; i = 2, \dots, n \quad (2.7)$$

$$\sum_{j=2}^n (x_{1jk} + x_{j1k}) = 2, \forall k = 1, \dots, m \quad (2.8)$$

$$x_{ijk} \in \{0, 1\}, \forall i = 1, \dots, n; j = 1, \dots, n; i \neq j; k = 1, \dots, m \quad (2.9)$$

$$y_{ik} \in \{0, 1\}, \forall i = 2, \dots, n; k = 1, \dots, m \quad (2.10)$$

$$v_{ik} \geq 0, \forall i = 2, \dots, n; k = 1, \dots, m \quad (2.11)$$

The objective is represented by Equation (2.1), which is to minimize the total distance traveled. Constraints (2.2) and (2.3) ensure that all customer demand is satisfied without violating vehicle capacity. Constraints (2.4) and (2.5) ensure flow conservation and that sub-tours are eliminated, respectively. Constraints (2.6) and (2.7) force the binary variables to be positive if material is delivered to node  $i$  on route  $k$ . Constraint (2.8) ensures that the depot is entered and exited on every vehicle route, and constraints (2.9) – (2.11) provide variable restrictions.

In recent work on the SDVRP, several researchers developed approaches for generating solutions to the SDVRP. Archetti et al. (2006) developed a Tabu search algorithm called

SPLITTABU to solve the SDVRP in which they showed that there always exists an optimal solution where the quantity delivered by each vehicle when visiting a customer is an integer number. Also, Archetti et al. (2008a) performed a mathematical analysis and proved that by adopting a SDVRP strategy, a maximum of 50% reduction can be achieved in the number of routes. Also they showed that when the demand variance is relatively small and the customer demand is in the range of 50% to 70% of the vehicle capacity, maximum benefits are achieved by splitting the customer's demand. Furthermore, Archetti et al. (2008b) presented a solution approach that combines heuristic search and integer programming. Boudia et al. (2007a) solved an SDVRP instance using a memetic algorithm with population management which produced better and faster results than the SPLITTABU approach (Archetti et al. (2006)). Mota et al. (2007d) proposed an algorithm based on scatter search methodology which generated excellent results compared to SPLITTABU.

Two approaches are used as a comparison with regard to this research. First, Jin et al. (2008) proposed a column generation approach to solve SDVRP with large demands, and in which the columns have route and delivery amount information and limited-search-with-bound algorithm is used to find the lower and upper bounds of the problem. They used column generation to find lower bounds and an iterative approach to find upper bounds for a SDVRP. They also suggested that their approach of solving the SDVRP does not yield good solutions for large customer demands and in such cases, they recommend solving the SDVRP instance as a CVRP. Second, Chen et al. (2007b) create test problems and developed a heuristic which is a combination of a mixed integer program and record-to-record travel algorithm to solve SDVRP.

Archetti and Sperenza (2012) have published an extensive survey on SDVRP and its variants. However, despite several exact optimization and metaheuristic solution methods being applied to the SDVRP, no previous research has applied the ant colony optimization metaheuristic to the SDVRP.

The number of customers for the 11 data sets from Jin et al. (2008) ranged from 50 to 100, with an additional node for the depot. The data sets also differ by amount of spare capacity per vehicle. The customers were placed randomly around a central depot and demand was generated randomly based on a high and low threshold. The number of customers for 21 data sets from Chen et al. (2007b) ranged from 8 to 288, with an additional node for the depot. The data sets do not have any spare vehicle capacity. The customers were placed on rings (i.e., circular pattern) surrounding a central depot and the demand was either 60 or 90, with a vehicle capacity of 100.

### 3. Ant Colony Optimization Approach

In this section I describe the ACO algorithm for SDVRP and in addition, I also provide some important literature relevant to the application of ACO to VRP and its variants.

Ant Colony Optimization (ACO) is a metaheuristic proposed by Dorigo (1992a). Inspired by foraging behavior of ants, ACO belongs to a class of metaheuristic algorithms that can be used to obtain near optimal solutions in reasonable computational time for combinatorial optimization problems. Ants communicate with one another by depositing pheromones, a trace chemical substance that can be detected by other ants (Rizzoli et al. (2004d). As ants travel, they deposit pheromones along their trail, and other ants tend to follow these pheromone trails. However during their journey, ants may randomly discover a new trail, which might be shorter or longer than the previous trail. Pheromones have a tendency to evaporate. Hence, over a period of time, the shortest trail (path) from the food source to the colony will have a larger amount of pheromone deposited as compared with other trails and will become the preferred trail.

The main elements in an ACO are ants that independently build solutions to the problem. For an ant  $k$ , the probability of it visiting a node  $j$  after visiting node  $i$  depend on the two attributes namely:

- **Attractiveness ( $\eta_{ij}$ ):** It is a static component that never changes. In the case of VRP, it is calculated as inverse of arc length for shortest path problems and for

other variants, it can depend on other parameters besides the arc length (e.g., in VRPTW it also depends on the current time and the time window limits of the customers to be visited (Rizzoli et al., 2004d)).

- **Pheromone trails( $\tau_{ij}$ ):** It is the dynamic component which changes with time. It is used to measure the desirability of insertion of an arc in the solution. In other words, if an ant finds a strong pheromone trail leading to a particular node, that direction will be more desirable than other directions. The trail desirability depends on the amount of pheromone deposited on a particular arc.

For solving a VRP, each individual ant simulates a vehicle. Starting from the depot, each ant constructs a route by selecting one customer at a time until all customers have been visited. Using the formula from Dorigo et al. (1997b), the ant selects the next customer  $j$  as shown in equation (2.12):

$$j = \begin{cases} \arg \max \{(\tau_{iu})(\eta_{iu}^\beta)\} & \text{for } u \notin M_k, q \leq q_0 \\ \text{Equation (2.13),} & \text{otherwise} \end{cases} \quad (2.12)$$

where  $\tau_{iu}$  is the amount of pheromone on arc  $(i,u)$ ,  $u$  being all possible unvisited customers. In classic VRP, locations already visited are stored in ants' working memory  $M_k$  and are not considered for selection. However, in the case of SDVRP, the locations for which the demands have not been fulfilled ( $demand > 0$ ) are stored in the ants' working memory and are considered for selection.  $\beta$  establishes correlation between the importance of distance with respect to the pheromone quantity ( $\beta > 0$ ).  $q$  is a randomly generated variable between 0 and 1 and  $q_0$  is a predefined static parameter. If equation (2.12) does not hold, the next customer to be visited is selected based on a random probability rule as shown in equation (2.13):

$$P_{ij} = \begin{cases} \frac{[(\tau_{ij})][(\eta_{ij}^\beta)]}{\sum_{j \in M_k} [(\tau_{ij})][(\eta_{ij}^\beta)]} & \text{if } j \notin M_k, q > q_0 \\ 0 \text{ (depot),} & \text{otherwise} \end{cases} \quad (2.13)$$

If the vehicle capacity constraint is satisfied, the ant will return to the depot before starting the next tour in its route. This selection process continues until all customers are visited by an ant. In ACO, the pheromone trail is updated locally during solution construction and globally at the end of construction phase. An interesting aspect of pheromone trail updating is that every time an arc is visited, its value is diminished which favors the exploration of other non-visited nodes and diversity in the solution. Pheromone trails are updated by reducing the amount of pheromone deposited on each arc  $(i,j)$  visited by an ant (local update). Also, after a predetermined number of ants construct feasible routes, pheromones are added to all the arcs of the best found solution (global update).

Local update on a particular arc  $(\tau_{ij})$  is updated done using equation (2.14) :

$$\tau_{ij} = (1-\alpha)\tau_{ij} + \alpha\tau_0 \quad (2.14)$$

where  $0 \leq \alpha \leq 1$  is the pheromone trail evaporation rate and  $\tau_0$  is the initial pheromone value for all arcs.

Global trial updating is done using equation (2.15):

$$\tau_{ij} = (1-\alpha)\tau_{ij} + \alpha L^{-1} \quad (2.15)$$

where  $L$  is the best found objective function value (total distance).

This procedure is repeated until a terminating condition is met. There is an another optional component called Daemon actions which are used to perform centralized actions such as calling a local search procedure or collecting global information to deposit addition pheromones on edges from a non-local perspective. Pheromone updates performed by daemons are called off-line pheromone updates.

The pseudo-code for ACO is shown below:

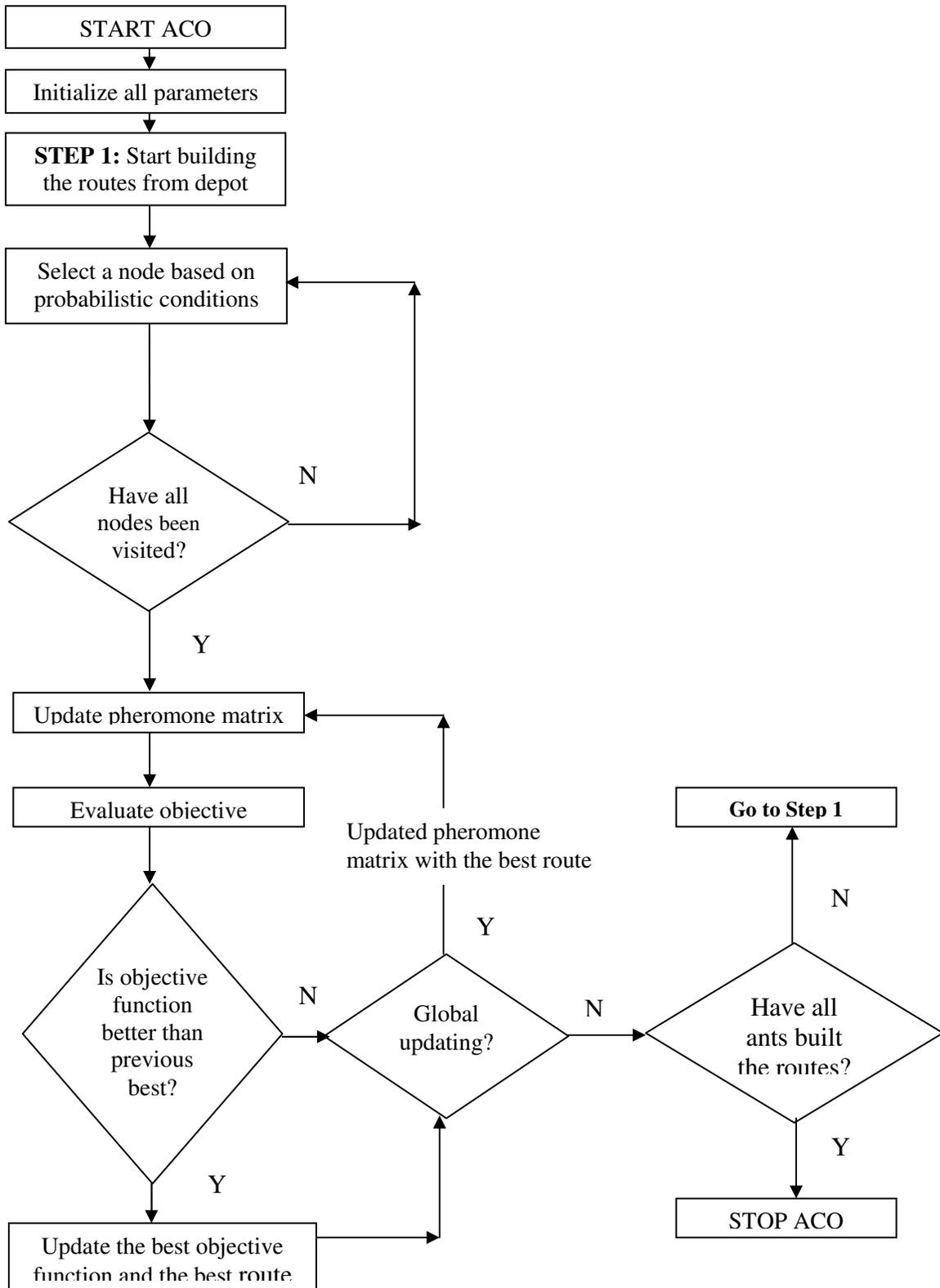
```

Procedure ACO
  While (terminating condition is not met)
    Generate_solutions ()
    Local_Update_of_Pheromones ()
    Global_Update_of_Pheromones ()

```

```
        Actions_If_Necessary () // this is optional  
    End while  
End procedure
```

The ACO flowchart is shown in Figure 2.1 below:



**Figure 2.1: ACO Flowchart**

Over a period of time, researchers have developed numerous ACO based solutions for VRP and its variants. One of the first papers on application of ACO in VRP was proposed by Bullheimer et al. (1997a; 1999a). They proposed a variant called “hybrid ACO” using 2-opt heuristic. Their algorithm was tested on fourteen Christofides benchmark problems and computation results showed that the results obtained were not as good as the ones obtained from other metaheuristics. Additionally, Gambardella et al. (1999b) proposed an algorithm based on ACO called MACS-VRPTW (Multiple Ant Colony System for Vehicle Routing Problems with Time Windows). This is the first paper in which a multi-objective minimization problem is solved using a multiple ant colony optimization algorithm. MACS-VRPTW not only provided improved solutions on benchmark test problems but also was on par or better than other existing methods in terms of solution quality and computation time. Next, Baran and Schaefer (2003) proposed a multi objective ACO for VRPTW based on MACS-VRPTW but instead of using two ant colonies, only one ant colony was used to find a set of Pareto optimal solutions for three objectives.

Rizzoli et al. (2004d) have done extensive surveys on ACO for VRP and its variants. Montemanni et al. (2004c) proposed an ACO solution called ACS-DVRP to solve the Dynamic VRP (DVRP) in which the large DVRP problem was divided into smaller static VRP problems. Bell et al. (2004a) proposed single and multiple ant colony methodologies to solve the VRP. Their experimental results showed that the best results were obtained when the candidate list size was between ten and twenty. Doerner et al. (2004b) proposed a parallel ant system algorithm for CVRP and this is the first paper which shows the effect of parallelization of processors on speed and efficiency. Additionally, Favaretto et al. (2007c) formulated and provided an ACO based solution for VRP with multiple time windows and multiple visits which consider periodic constraints. Computation results show that their proposed algorithm provides better solutions as compared to some of the other metaheuristics published in the literature. Also, Gajpal and Abad (2009) proposed an ant colony system for VRP with simultaneous delivery and pickup (VRPSDP). Computational results on benchmark test problems show that the

proposed algorithm provides better results both in terms of solution quality and CPU time. Finally, Hu et al. (2011) provided an ACO based solution for distributed planning problems for home delivery in which a revised methodology to update the pheromone and the probability matrix is proposed.

However, to the best of my knowledge and despite previous success applying ACO to variants of the VRP, no journal article has applied ACO to the SDVRP and experimentally tested the ability of the algorithm on SDVRP instances.

#### **4. Computational experiments**

One of the route improvement strategies is to have a candidate list to determine the next location for each customer. Only a set of predetermined closest locations are included in the candidate list. In previous research (Bullnheimer et al. (1999a)), irrespective of the problem size, the size of the candidate list was set to one fourth of the total number of customers. In pilot testing, I experimented with different candidate list sizes and for our research the candidate list size of one ninth ( $n/9$ , where  $n$  is the number of customers) was found to yield the best solutions. Additionally, in the case of CVRP, an ant (vehicle) travels to a customer (node) only if the customer's demand can be completely fulfilled with the remaining vehicle capacity. But in the case of SDVRP, since a customer's demand can be split amongst multiple vehicle routes, the ant travels to a customer based on three conditions: (1) If the customer is in the candidate list, (2) if the customer's demand is not completely fulfilled, and (3) there is remaining capacity on the vehicle. If the above conditions cannot be satisfied for any location, the ant (vehicle) returns to the depot.

The ACO algorithm for this study was coded in Java on a Windows7, Intel i5 2.4 Ghz, 4 GB RAM computer. For all our test datasets, search parameters were tuned during pilot-testing and set as shown in Table 2.1. The algorithm was tested against two procedures from the literature, namely Jin et al. (2008) and Chen et al. (2007b). Each problem in the dataset was run in 10 separate iterations (Fuellerer et al. (2010a)). The results are shown

in Table 2.2 and Table 2.3. The vehicle capacity for datasets in Table 2.2 and Table 2.3 are 160 and 100 respectively.

**Table 2.1: Parameters**

Parameter	Values
$\alpha$	<b>0.5</b>
$\beta$	1.3
$\tau_0$	<b>10-5</b>
$q_0$	0.9
<b>m (global update counter)</b>	<b>10</b>
<b>Number of iterations</b>	100,000

**Table 2.2: Comparing ACO results versus Jin et al. (2008)**

Dataset	Ant Colony Optimization				Results from Jin et al.		
	Objective Function (Average (std dev))	Objective Function (Best)	Best Time(s)	Total Time(s)	Objective Function	Total Time(s)	GAP
s51d2	744.03(14.07)	727.28	186.59	699.56	<b>722.93</b>	10741	0.60%
s51d3	1001.97(15.87)	982.66	164.5	843.23	<b>968.85</b>	833	1.43%
s51d4	1654.56(12.68)	1629.09	1053.95	1074.66	<b>1605.64</b>	789	1.46%
s51d5	1416.60(20.37)	1389.01	519.44	1015.48	<b>1361.24</b>	10	2.04%
s51d6	2302.72(14.16)	2267.97	584.65	1339.20	<b>2196.35</b>	478	3.26%
576d2	1161.19(12.47)	<b>1134.27</b>	<b>1431.9</b>	1742.09	1146.68	75074	<b>-1.08%</b>
s76d3	1527.25(19.06)	1502.36	979	2078.88	<b>1474.89</b>	3546	1.86%
s76d4	2218.51(21.63)	2191.83	337.7	1310.30	<b>2157.87</b>	369	1.57%
s101d2	1484.12(16.99)	<b>1457.39</b>	<b>930.81</b>	3352.49	1460.54	189392	<b>-0.22%</b>
s101d3	2000.94(33.52)	<b>1948.09</b>	<b>3166.21</b>	3938.37	1956.91	36777	<b>-0.45%</b>
s101d5	2972.54(17.29)	2945.41	3778.25	4947.82	<b>2885</b>	5043	2.09%

\*The objective function values highlighted in bold are the best results

Note: GAP indicates ACO versus best known solution. A negative GAP indicates a new best solution when compared to previous literature.

**Table 2.3: Comparing ACO results versus Chen et al. (2007a)**

Dataset	Ant Colony Optimization				Results from Chen et al.		
	Objective Function (Average (std dev))	Objective Function (Best)	Best Time (s)	Total Time(s)	Objective Function	Time(s)	GAP
sd1	240(0)	240	1.743	76.01	<b>228.28</b>	0.7	5.13%
sd2	758(11.35)	740	56.77	87.25	<b>714.4</b>	54.4	3.58%
sd3	451.52(2.42)	447.69	66.12	81.81	<b>430.61</b>	67.3	3.97%
sd4	679.04(1.86)	673.89	65.43	202.75	<b>631.06</b>	400	6.79%
sd5	1454.91(3.85)	1445.64	106.92	405.28	<b>1408.12</b>	402.7	2.66%
sd6	860.45(0)	860.45	0.13	378.08	<b>831.21</b>	408.3	3.52%
sd7	3640(0)	<b>3640</b>	<b>0.3</b>	603.01	3714.4	403.2	<b>-2.00%</b>
sd8	5110.80(45.67)	<b>5068.28</b>	<b>214.58</b>	963.57	5200	404.1	<b>-2.53%</b>
sd9	2140.15(14.99)	2129.59	201.15	1017.24	<b>2059.84</b>	404.3	3.39%
sd10	2841.07(14.97)	2807.05	1352.83	2013.42	<b>2749.11</b>	400	2.11%
sd11	13280(0)	<b>13280</b>	<b>2.65</b>	3086.07	13612.12	400.1	<b>-2.44%</b>
sd12	7280.06(0)	<b>7280.06</b>	<b>2337.17</b>	3367.17	7399.06	408.3	<b>-1.61%</b>
sd13	10281.74(282.23)	<b>10171.92</b>	<b>4653.16</b>	5232.16	10367.06	404.5	<b>-1.88%</b>
sd14	11069.11(46.97)	<b>11021.54</b>	<b>7325.6</b>	9208.81	11023	5021.7	<b>-0.01%</b>
sd15	15405.92(79.36)	15309.9	12816.82	17594.98	<b>15271.77</b>	5042.3	0.25%
sd16	3411.31(11.17)	<b>3398.69</b>	<b>0.743</b>	17201.99	3449.05	5014.7	<b>-1.46%</b>
sd17	26586.11(16.56)	<b>26560.11</b>	<b>12188.12</b>	23866.41	26665.76	5023.6	<b>-0.40%</b>
sd18	14772.57(30.52)	14720.11	24301.78	24439.43	<b>14546.58</b>	5028.6	1.19%
sd19	20376.31(29.96)	<b>20312.44</b>	<b>11455.71</b>	38677.42	20559.21	5034.2	<b>-1.20%</b>
sd20	40479.27(51.83)	<b>40390.68</b>	<b>49658.4</b>	78854.50	40408.22	5053	<b>-0.043%</b>
sd21	11449.88(26.31)	<b>11411.61</b>	<b>1.64</b>	121148.80	11491.67	5051	<b>-0.70%</b>

\*The objective function values highlighted in bold are the best results

Note: GAP indicates ACO versus best known solution. A negative GAP indicates a new best solution when compared to previous literature.

The GAP column in Table 2.2 and Table 2.3 is the percentage difference in objective function values of ACO and those obtained from Jin et al. (2008) and Chen et al. (2007b) respectively. From Table 2.2, ACO solutions were between 0.6% - 3.26% of the objective function values from Jin et al. (2008) but the computational times were much faster. Also for 3 datasets, ACO found the best known solutions. For example, in problem s76d2, I found an improved solution that is 1.08% better than the previously best known solution. This problem is a 75 node problem and is one of three problems that the best known solution was improved on in this dataset using the ACO methodology.

However, much greater success was found in improving the best known solutions in the problem sets of Chen et al. (2007a). From Table 2.3, for 11 out of the 21 datasets, ACO produced better results; however this often came at the expense of computational time. For example in problem sd8, ACO was able to find the objective function value 5068.28. This value is 2.53% better than the previously known best solution. Overall, ACO was able to find improved solutions in eleven of the problems that ranged from 0 to 2.53% in improvement. However, for several of the smaller problems (sd1-sd5), the method appeared to have difficulty. Since these problems consist of fewer than 40 nodes, it was expected that the combination of using a candidate list size of  $n/9$  and the small problem size may have restricted the algorithm from considering enough nodes in the route construction process.

Therefore, in post-hoc testing of these 5 datasets, the candidate list size was removed in order to assess the ability of ACO to solve these smaller problems without the need for a candidate list size. The results of this post-hoc test are listed in Table 2.4. Notice that after the candidate list was removed, the objective function for sd1 was improved from 240 to 228.28, which is equal to the previously best known solution. Also, as you can see from Table 2.3 and Table 2.4, for datasets sd2, sd3 and sd4, a significant improvement in objective function values at the expense of computational time were obtained without using a candidate list.

**Table 2.4: Post-hoc results (without using a candidate list)**

Ant Colony Optimization					Results from Chen et al.		
Dataset	Objective Function (Average (std dev))	Objective Function (Best)	Best Time (s)	Total Time(s)	Objective Function	Time(s)	GAP
sd1	228.28(0)	<b>228.28</b>	0.25	27.27	228.28	0.7	<b>0.00%</b>
sd2	747.56(8.86)	734.34	92.53	121.29	<b>714.4</b>	54.4	2.79%
sd3	454.72(6.9)	440.07	48.56	111.11	<b>430.61</b>	67.3	2.20%
sd4	670.18(3.93)	665.94	131.68	270.08	<b>631.06</b>	400	5.53%
sd5	1454.49(4.32)	1448.01	261.28	535.34	<b>1408.12</b>	402.7	2.83%

\*The objective function values highlighted in bold are the best results

As seen from the results Table (Table 2.2 and Table 2.3), ant colony optimization has the ability to produce results within only a few percent of the optimal solutions. Also, SDVRP has complex constraints that the memory and learning features of ACO are able to navigate and find improved solutions to, consistent with previous research on other variants of the VRP. In our experimental results, for larger problem instance (Table 2.3), ACO produced better results than the optimal solutions but at the expense of computational time. Also, the use of candidate lists on larger problems and tuning of ACO parameters significantly improves the ability of ACO to find better solutions.

The objective function values for the two datasets are compared with the dual bound obtain by column generation (working paper, Wilck and Cavalier, 2012a), results of which are shown in Table 2.5 and Table 2.6 respectively. The GAP represents the percentage difference between the objective function values of ACO and the column generation dual bound. As you can see from Table 2.5 and Table 2.6 below, the percentage difference between ACO objective function and column generation dual bound ranges from 0 % to 6.36 % (2007a) and 3.60% to 8.17%(2008) respectively.

**Table 2.5: Comparison of ACO objective function for Chen et al. (2007a) and Column generation dual bound (Working paper, Wilck and Cavalier)**

Dataset	ACO Objective function	Column generation dual bound*	GAP
sd1	240	228.28	4.88%
sd2	740	708.28	4.29%
sd3	447.69	430.58	3.82%
sd4	673.89	631.05	6.36%
sd5	1445.64	1390.57	3.81%
sd6	860.45	831.21	3.40%
sd7	3640	3640.00	0.00%
sd8	5068.28	5068.28	0.00%
sd9	2129.59	2044.23	4.01%
sd10	2807.05	2684.84	4.35%
sd11	13280	13265.29	0.11%
sd12	7280.06	7275.97	0.06%
sd13	10171.92	10093.72	0.77%
sd14	11021.54	10632.67	3.53%
sd15	15309.9	15146.92	1.06%
sd16	3398.69	3375.95	0.67%
sd17	26560.11	25320.09	4.67%
sd18	14720.11	14253.94	3.17%
sd19	20312.44	19768.23	2.68%
sd20	40390.68	38071.58	5.74%
sd21	11411.61	11062.32	3.06%

*\*Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.*

*Column Generation stopping criteria: 5% GAP [i.e.,  $GAP = (Primal\ Solution - Dual\ Bound) / Primal\ Solution$ ].*

**Table 2.6: Comparison of ACO objective function for Jin et al. (2008) and Column generation dual bound (Working paper, Wilck and Cavalier)**

Dataset	ACO Objective function	Column generation dual bound*	GAP
s51d2	727.28	688.83	5.29%
s51d3	982.66	920.58	6.32%
s51d4	1629.09	1520.71	6.65%
s51d5	1389.01	1310.12	5.68%
s51d6	2267.97	2115.20	6.74%
576d2	1134.27	1093.39	3.60%
s76d3	1502.36	1399.37	6.86%
s76d4	2191.83	2039.11	6.97%
s101d2	1457.39	1395.25	4.26%
s101d3	1948.09	1859.36	4.55%
s101d5	2945.41	2704.63	8.17%

*\*Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.*

*Column Generation stopping criteria: 5% GAP [i.e.,  $GAP = (Primal\ Solution - Dual\ Bound) / Primal\ Solution$ ].*

## 5. Conclusions and Future directions

In this study, I presented an ACO based approach to solve the Split Delivery Vehicle Routing Problem (SDVRP). The algorithm was tested on benchmark test problems and results obtained were promising. Also for some instances, the best known solution to date was found using the ACO algorithm. Also, an interesting observation that I can highlight and consider for future research is the use of a candidate list size. As mentioned in previous literature (1999a), a candidate list size of one fourth of the total number of customers is recommended but for my datasets, a candidate list of one ninth the total number of customers was found to yield better results during pilot testing. However, at times, this restricted the ability to find improved solutions on the smallest problems. Hence, further research on developing a logic that will generate an ideal candidate list based on total number of customers is needed. Also in the future, I hope to focus on improving the ACO algorithm for SDVRP by (1) using local exchange heuristics to improve the solution, and (2) using specialized groups of ants and multiple colonies as mentioned in the literature Bell and McMullen (2004a), Gambardella et al. (1999b), and others.

## 6. References

- Aleman, R.E., Zhang, X., & Hill, R. R. (2010b). An adaptive memory algorithm for the split delivery vehicle routing problem, *Journal of heuristics*, 16(3), 441-473.
- Archetti, C., Savelsbergh, M., & Hertz, A. (2006). A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem. . *Transportation Science*, 40(1), 64-73.
- Archetti, C., Savelsbergh, M., & Speranza, M. G. (2008a). To split or not to split: That is the question. *Transportation Research, Part E* 44(1), 114-123.
- Archetti, C., & Speranza, M. G. (2012). Vehicle routing problems with split deliveries. [10.1111/j.1475-3995.2011.00811.x]. *International Transactions in Operational Research*, 19(1-2), 3-22.
- Archetti, C., Speranza, M. G., & Savelsbergh, M. (2008b). An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem. *Transportation Science*, 42(1), 22-31.
- Barán, B., & Schaerer, M. (2003). *A multiobjective ant colony system for vehicle routing problem with time windows*. Paper presented at the Proceedings of the 21st IASTED International Conference Applied Informatics, Austria, 97-102.
- Bullnheimer, B., Hartl, R.F., & Strauss, C. (1997a). *Applying the ant system to the vehicle routing problem*. Paper presented at the In Proceedings of the 2nd International Conference on Metaheuristics -MIC97 INRA Sophia-Antipolis & PRiSM, Versailles.
- Bullnheimer, B., Hartl, R.F., & Strauss, C. (1999a). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89, 319-328.
- Bell, J. E., & McMullen, P. R. (2004a). Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18, 41-48.
- Boudia, M., Prins, C., & Reghioui, M. (2007a). *An effective memetic algorithm with population management for the split delivery vehicle routing problem*. Paper presented at the Proceedings of the 4th international conference on Hybrid metaheuristics, Heidelberg

- Chen, S., Golden, B., & Wasil, E. (2007b). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4), 318-327.
- Doerner, K., Hartl, R. F., Kiechle, G., Lucka, M., & Reimann, M. (2004b). *Parallel ant systems for the capacitated vehicle routing problem*. . Paper presented at the Evolutionary Computation in Combinatorial Optimization: 4th European Conference, EvoCOP 2004, Berlin.
- Dorigo, M. (1992a). *Ph.D. Thesis Optimization, learning and natural algorithms (in Italian)*. Politecnico di Milano, Italy.
- Dorigo, M., Gamberdella, L.M. (1997b). Ant colonies for traveling salesman problem. *BioSystem*, 43(1), 73-81.
- Dror, M. & Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, 23, 141-145.
- Dror, M., Laporte, G. & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3), 239-254.
- Dror, M., & Trudeau, P. (1990). Split Delivery Routing. *Naval Research Logistics* 37, 383-402.
- Favaretto, D., Moretti, E., & Pellegrini, P. (2007c). Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics*, 10(2), 263-284.
- Frizzell, P., & Giffin, J. (1992b). The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research*, 9, 101-116.
- Fuellerer, G., Doerner, K. F., Hartl, R. F., & Iori, M. (2010a). Metaheuristics for vehicle routing problems with three-dimensional loading constraints. *European Journal of Operational Research*, 201(3), 751-759. doi: 10.1016/j.ejor.2009.03.046
- Gajpal, Y., & Abad, P. (2009). An ant colony system(ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36, 3215-3223.

- Gambardella, L. M., Taillard, E., Agazzi, G. (1999b). MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In D. Corne, M. Dorigo & F. Glover (Eds.), *New Ideas in Optimization* (pp. 63-76). UK: McGraw-Hill.
- Hu Meng-Jie, Wang Jian-Ping, & Xiao-Min, L. (2011). Solutions to VRP in Home Delivery Based on Ant Colony Optimization Algorithm. *Advanced Materials Research, 159*, 100-104.
- Jin, M., Liua, K., & Eksioglu, B. (2008). A column generation approach for the split delivery vehicle routing problem. *Operations research letters, 36*(2), 265-270.
- Montemanni, R., Gambardella, L. M., Rizzoli, A.E., & Donati. A.V. (2004c). A new algorithm for a dynamic vehicle routing problem based on ant colony system. *Technical Report TR-23-02, IDSIA, Galleria 2. Manno, 6928, Switzerland.*
- Mota, E., Campos, V., & Corberán, Á. (2007d). A New Metaheuristic for the Vehicle Routing Problem with Split Demands. *Lecture Notes in Computer Science, 4446* 121-129.
- Rizzoli, A. E., Oliverio, F., Montemanni, R., & Gambardella, L. M. (2004d). Ant colony optimisation for vehicle routing problem: from theory to applications. *Technical Report TR-15-04.*
- Wilck, J. H., & Rajappa, G. (2010c). *Ranking Construction Heuristic Solutions for a Hybrid Genetic Algorithm for the Split Delivery Vehicle Routing Problem (SDVRP)*. Paper presented at the INFORMS Southern Regional Conference 2010, Huntsville, AL.
- Wilck, J.H. & Cavalier, T.M. (2012a). A Column Generation Method for the Split Delivery Vehicle Routing Problem using a Route-Based Formulation, Working Paper, University of Tennessee, Knoxville and Pennsylvania State University.

**CHAPTER III**  
**A HYBRID GENETIC ALGORITHM APPROACH TO SOLVE THE**  
**SPLIT DELIVERY VEHICLE ROUTING PROBLEM**

## Publication Statement

This paper is a joint work between Gautham P. Rajappa and Dr. Joseph H. Wilck. We are currently working on this paper for publication.

## Chapter Abstract

Vehicle Routing Problem (VRP) is a combinatorial optimization problem in the field of transportation and logistics. There are various variants of VRP which have been developed over the years one of which is the Split Delivery Vehicle Routing Problem (SDVRP). The SDVRP allows customers to be assigned to multiple routes. A hybrid genetic algorithm comprising a combination of Ant Colony Optimization, genetic algorithm and heuristics is proposed and tested on benchmark SDVRP test problems.

## 1. Introduction

Vehicle Routing Problem (VRP) is an important combinatorial optimization problem in the field of transportation and logistics. The objective of the VRP is to minimize the cost associated with delivering goods to a set of customers with known demands with vehicle routes originating and terminating at a central depot or depots. The basic underlying concept of a VRP is derived from Traveling Salesman Problem (TSP) but instead of a single route, VRP extends TSP to multiple routes in which a set of customers are serviced in a particular route with the objective of minimizing the total cost. VRP was first proposed by Dantzig and Ramser (1959) to reduce costs in distributing gasoline from a central depot to various bunks. Over a period of time, various variants of VRP were developed, a brief description of which is given below:

- **Vehicle Routing Problem with Time Windows (VRPTW):** The customer location has a time frame within which the deliveries have to be made.
- **Capacitated Vehicle Routing Problem (CVRP):** In this case, there is a restriction on the delivery vehicle capacity
- **Split Delivery Vehicle Routing Problem (SDVRP):** It is a relaxed version of CVRP in which the goods can be delivered to the customer by more than one route (vehicle).
- **Multiple Depot Vehicle Routing Problem (MDVRP):** Customers are served from multiple depot.

- **Vehicle Routing Problem with Pick-Ups and Deliveries (VRPPD):** In this case, the delivery vehicle picks up goods from a pick-up locations and drops it off at the customer location
- **Vehicle Routing Problem with Backhauls (VRPB):** In the case, once all the deliveries are done to the customer, the vehicle needs to pickup goods from the customer.
- **Periodic Vehicle Routing Problem (PVRP):** In this case, the deliveries are done in days.
- **Stochastic Vehicle Routing Problem (SVRP):** In this case, the components of the problem are stochastic in nature.

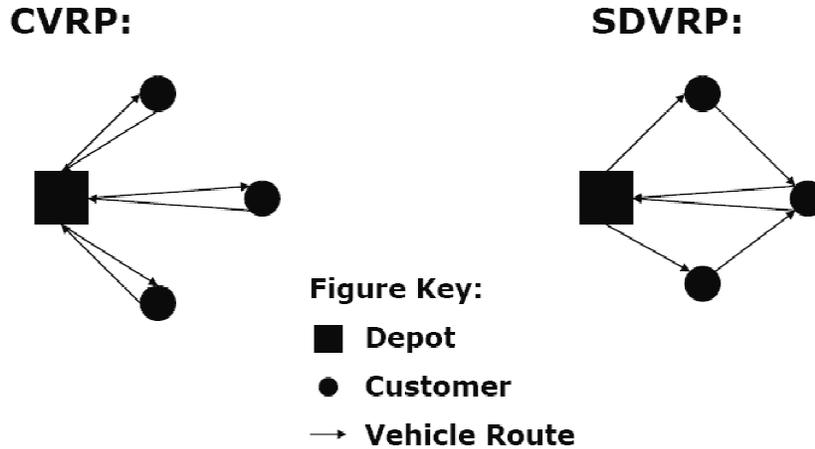
The objective of this paper is on Split Delivery Vehicle Routing Problem (SDVRP). This paper focuses on developing a hybrid genetic algorithm to solve SDVRP. Due to constraints of the problem, a pure genetic algorithm cannot be applied to generate a new set of feasible solutions and hence the name, hybrid genetic algorithm. In this paper, I use a combination of Ant Colony Optimization, heuristics and Genetic Algorithms to solve the split delivery vehicle routing problem.

The rest of the chapter is organized as follows. Section 2 provides an overview of SDVRP. Section 3 focuses on literature of various methodologies that have been developed to solve the SDVRP. Section 4 explains the proposed hybrid genetic algorithm in detail. Computation experiments are discussed in Section 5 and conclusions and future work is discussed in Section 6. Also, for details about Ant colony optimization, please refer to Chapter I and Chapter II of the dissertation.

## **2. Split Delivery Vehicle Routing Problem (SDVRP)**

SDVRP was first developed Dror and Trudeau (1989a; 1990) as a relaxed version of CVRP. They developed a heuristic algorithm to solve the problem and also proved that when triangular inequality i.e. sum of two sides of a triangle is greater than the third side holds good, an optimal solution exists in the SDVRP in which two routes cannot have more than one common customer. They also showed that SDVRP is NP-hard. As shown

in Figure 3.1 below, in the case of a CVRP, each customer is served by only one vehicle but since SDVRP is a relaxed version of CVRP, the customer demand can be split between vehicles.



**Figure 3.1: CVRP v/s SDVRP**

Consider for example, the customer demand is 300 and the vehicle capacity is 100. In the case of CVRP, we require three vehicles but in the case of SDVRP, since the customer demand can be split amongst multiple vehicles, we just require 2 vehicles to fulfill the customer demand. SDVRP has potential in savings in terms of the distance traveled as well as the number of vehicles used.

According to Aleman et al. (2010d), the SDVRP is defined on an undirected graph  $G = (V, E)$  where  $V = \{0, 1, \dots, n\}$  is the set of  $n + 1$  nodes of the graph, and  $E = \{(i, j) : i, j \in V, i < j\}$  is the set of edges connecting the nodes. Node 0 represents a depot where a fleet  $M = \{1, \dots, m\}$  of identical vehicles with capacity  $Q$  are stationed, while the remaining node set  $N = \{1, \dots, n\}$  represents the customers. A non-negative cost, usually a function of distance or travel time,  $c_{ij}$  is associated with every edge  $(i, j)$ . Each customer  $i \in N$  has a demand of  $q_i$  units. The optimization problem is to determine which customers are served by each vehicle and what route will the vehicle follow to serve those assigned customers, while minimizing the operational costs of the fleet, such as

travel distance, gas consumption, and vehicle depreciation. Various problem formulations for SDVRP have been developed over the years and the most frequently used formulations are from Dror and Treadeau (1990), Frizzell and Giffin (1992b), Dror et al (1994a) which can be found in the literature.

For a detailed mathematical model formulation of SDVRP, please refer to Section 2 of Chapter II.

### **3. Literature Review**

In this section, an extensive literature review on various methodologies that have been developed to solve the SDVRP is conducted. Both exact and heuristic methods have been proposed by various researchers to solve SDVRP. For large problem instances, it's not convenient to solve SDVRP using exact approaches due to large computational cost and hence, heuristic approach is the only way to obtain near-optimal solutions. SDVRP was introduced by Dror and Trudeau (1989a) in the year 1989. In their paper they showed that if the demand is relatively low to the vehicle capacity and the triangular inequality holds good (i.e. sum of two sides of a triangle is greater than the third side holds good, an optimal solution exists in the SDVRP in which two routes cannot have more than one common customer), there is little benefit of splitting the demands. In contrast, if the customer demand is at least 10% more than the vehicle capacity, the overall cost associated with SDVRP is lower as compared to that of a regular VRP. Sierksma and Tijssen (1998c) proposed a set-covering formulation for the SDVRP to build the helicopters schedule for supporting offshore platforms in the North Sea to exchange crews. Archetti et al. (2008a) performed a mathematical analysis and proved that by adopting a SDVRP strategy, a maximum of 50% reduction would be achieved in the number of routes. Also they showed that when the demand variance is relatively small and the customer demand is in the range of 50% to 70% of the vehicle capacity, maximum benefits can be achieved by splitting the customer's demand.

Archetti et al. (2006b) developed a Tabu search algorithm called SPLITTABU to solve the SDVRP in which they showed that always exists an optimal solution where the quantity delivered by each vehicle when visiting a customer is an integer number. In the paper on an optimization based heuristics for SDVRP, Archetti et al. (2008b) present a solution approach that combines heuristics search and integer programming. The IP is used to investigate the search space identified initially by a Tabu search heuristics. Boudia et al. (2007b) solved an SDVRP instance using memetic algorithm with population management which produced better and faster results than the SPLITTABU approach (Archetti et al., 2006b).

Mota et al. (2007d) proposed an algorithm based on scatter search methodology with the objective function of having minimum number of vehicles. For customer demands less than half of the vehicle capacity, their results were found to be excellent as compared to the results obtained by SPLITTABU proposed by Archetti et al. (2006b). But for demand over half the vehicle capacity, their results were not good. Mullaseril et al. (1997b) modeled a feed distribution problem in a cattle ranch in Arizona as SDVRP with time windows to schedule a fleet of trucks to distribute feed to cattle in various pens spread across the ranch.

Nakao and Nagamochi (2007e) proposed a dynamic program based heuristics to solve a Discrete Split Delivery Vehicle Routing problem. A Discrete SDVRP is a variant of SDVRP in which each customer demand may have more than one item, each of which cannot be split where items may have more than one size. Jin et al. (2008d) proposed a column generation approach to solve SDVRP with large demands in which the columns have route and delivery amount information and limited-search-with-bound algorithm is used to find the lower and upper bounds of the problem. They used a column generation to find lower bounds and an iterative approach to find upper bounds for a SDVRP. They also suggested that their approach of solving the SDVRP does not yield good solutions for large customer demands and in such cases, they recommend solving the SDVRP instance as a CVRP.

Aleman et al. (2010d) proposed three heuristic approaches to solve the SDVRP. The first approach is an adaptive constructive algorithm called route angle control measure, which yielded good results for large customer demands problem. The second approach is an iterative approach which solves the adaptive constructive algorithm repeatedly. The third approach was a variable neighborhood descent which produced the best results amongst all the three approaches. These algorithms provided better results than other approaches on benchmark test problems. Chen et al. (2007c) developed a heuristic that combines a mixed integer program and record-to-record travel algorithm to solve SDVRP.

Moghaddam et al. (2007f) used simulated annealing to solve SDVRP with the objective function of maximizing the vehicle utilization. Ambrosino and Sciomachen (2007a) proposed a SDVRP solution based on clustering procedure along with a local search to solve a food distribution problem for a Italian company.

## **4. Hybrid Genetic Algorithm Approach**

### ***4.1 Genetic Algorithms***

Genetic algorithms are population based search algorithms to solve combinatorial optimization problems. It was first proposed by John Holland (1989b). In these algorithms the search space (population) of a problem is represented as a collection of individuals (chromosomes). Genetic algorithms generate solutions for optimization problem based on theory of evolution using concepts such as reproduction, crossover and mutation. The fundamental concept of a genetic algorithm states a set of conditions to achieve global optima. These conditions describe the reproduction process and ensure that better solution remain in future generations and weaker solutions be eliminated from future generations. This is similar to the Darwin's survival of fittest concept in the theory of evolution. A typical genetic algorithm consists of the following steps (1989b):

- ***Step 1:*** *Generate an initial population of  $N$  solutions.*
- ***Step 2:*** *Evaluate each solution of the initial population using a fitness function/objective function.*

- **Step 3:** *Select solutions as parents for the new generation based on probability or randomness. The best solutions (in terms of fitness or objective) have a higher probability of being selected than poor solutions.*
- **Step 4:** *Use the parent solutions from Step 3 to produce the next generation (called offspring). This process is called as crossover. The offspring are placed in the initial set of solutions replacing the weaker solutions.*
- **Step 5:** *Randomly alter the new generation by mutation. Usually this is done using a mutation probability.*
- **Step 6:** *Repeat Steps 2 through 5 until a stopping criteria is met.*

Thus the genetic algorithm search mechanism consists of three phases: (1) Evaluation of fitness function of each solution in the population (2) selection of parent solutions based on fitness values and (3) application of genetic operations such as crossover and mutation to generate new offspring. For additional descriptions of genetic algorithms, please refer to Chapter I.

Due to the constraints of a SDVRP, it is not possible to directly use genetic algorithm in the way it is described above. In particular, after crossover and mutation, there may be solutions which do not satisfy the constraints. Hence, to obtain a feasible set of offspring, we may need to modify the way crossover is done or another possibility is to remove infeasible solutions after mutation and replace them with the solutions having higher fitness value in the old population (2002b). Hence a hybrid genetic algorithm needs to be developed to ensure feasibility in the new generation.

The hybrid genetic algorithm is described below:

- **Solution encoding:** It's represents a feasible vehicle route. The solutions are encoded as a series of random numbers from 0 to N, wherein, each N represents a node (customer location) and 0 represents a depot. For example, a route is represented as [0,1,2,3,0,3,4,5,0].

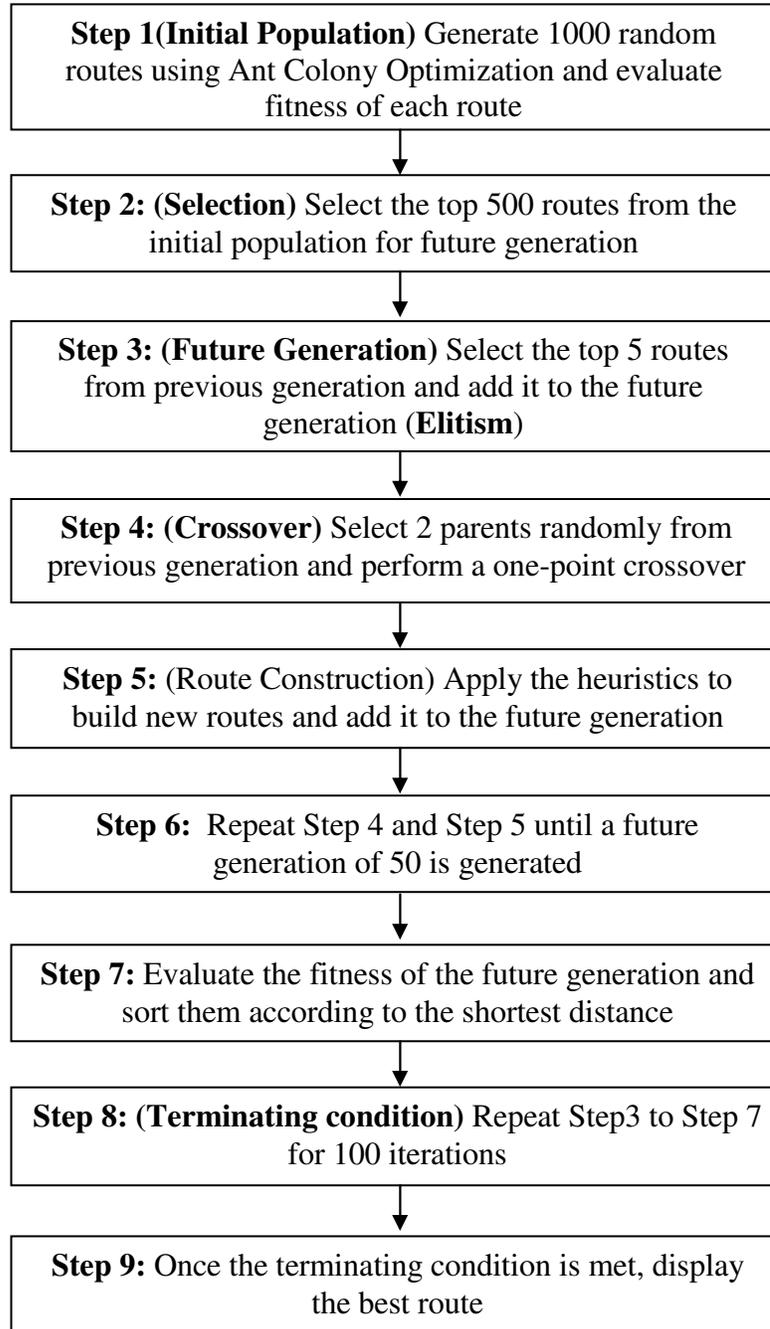
- **Initial population:** The initial population in the genetic algorithm is normally generated randomly but other approaches such as heuristics approach and ant colony optimization can also be applied to get a good set of initial population. For the hybrid genetic algorithm, 1000 random solutions from ant colony optimization are used for initial population.
- **Fitness:** The objective function is evaluated for each route from the initial population and then a corresponding fitness value is assigned. The fitness value is the total distance of a particular route.
- **Selection:** Using the fitness value of each route, the top 500 routes from the initial population are selected for future generation.
- **Future Generation (Crossover and mutation):**
  - The size of the future generation is set to 50.
  - Due to the constraints of SDVRP, mutation was not considered.
  - **Elitism:** The top 5 results from previous generation were used in the next generation
  - **Crossover:** Two parents are randomly selected from the previous generation. A one point crossover is then applied to each of these parents to generate future generation using the heuristics described below. Crossover is performed until 50 new routes are generated.
- **Heuristics:** The routes are constructed as follows:
  - **Condition 1:** For all the available nodes (demand is not satisfied), add the next node to the route if:
    - The node's demand is less than the remaining capacity of the vehicle and
    - The next node is closest to the previous node and
    - The next node has the largest demand amongst all the nodes.
  - **Condition 2:** If condition 1 is not satisfied, then for all the available nodes (demand is not satisfied), add the next node to the route if:
    - The node's demand is less than the remaining capacity of the vehicle and

- The next node is closest to the previous node.
- If condition 1 and condition 2 are not satisfied, go back to the depot.
- **Termination condition:** For 100 iterations, repeat the Fitness to Heuristics procedure and then display the best route.

The flowchart for the hybrid genetic algorithm is shown in Figure 3.2 below:

## **5. Computation experiments**

The Hybrid genetic algorithm for this study was coded in Java on a Windows7, Intel i5 2.4 Ghz, 4 GB RAM computer. For all our test datasets, the algorithm parameters were tuned during pilot-testing and set as shown in Table 3.1 below. The algorithm was tested on two datasets from the literature, namely Jin et al.(2008d) and Chen et al. (2007c) , and the comparative results are shown below in Table 3.2 and Table 3.3 respectively. The vehicle capacity for datasets in Table 3.2 and Table 3.3 are 160 and 100 respectively.



**Figure 3.2: Hybrid GA Flowchart**

**Table 3.1: Parameters**

Parameter	Values
Initial Population	500
Size of Future Generation	50
Elite List	5
Number of future generation (Terminating condition)	100

**Table 3.2: Comparing Hybrid GA results versus Jin et al.(2008d)**

Dataset	Hybrid Genetic Algorithm			Results from Jin et al.		
	Objective Function (Average (std dev))	Objective Function (Best)	Total Time (s)	Objective Function	Total Time(s)	GAP
<b>s51d2</b>	862.67(11.44)	845.86	2.22	<b>722.93</b>	10741	17.00%
<b>s51d3</b>	1118.48(23.45)	1080.32	2.409	<b>968.85</b>	833	11.51%
<b>s51d4</b>	1775.10(15.90)	1752.79	2.642	<b>1605.64</b>	789	9.16%
<b>s51d5</b>	1542.91(14.17)	1512.46	2.52	<b>1361.24</b>	10	11.11%
<b>s51d6</b>	2401.90(1.20)	2398.47	2.884	<b>2196.35</b>	478	9.20%
<b>s76d2</b>	1292.75(5.64)	1282.8	4.2	<b>1146.68</b>	75074	11.87%
<b>s76d3</b>	1674.94(14.12)	1649.51	4.6	<b>1474.89</b>	3546	11.84%
<b>s76d4</b>	2396.14(24.93)	2357.02	4.87	<b>2157.87</b>	369	9.23%
<b>s101d2</b>	1624.82(20.89)	1586.97	7.26	<b>1460.54</b>	189392	8.66%
<b>s101d3</b>	2158.10(24.09)	2122.04	7.94	<b>1956.91</b>	36777	8.44%
<b>s101d5</b>	3134.49(17.22)	3109.88	8.55	<b>2885</b>	5043	7.79%

**Table 3.3: Comparing Hybrid GA results versus Chen et al. (2007c)**

Dataset	Hybrid Genetic Algorithm			Results from Chen et al.		
	Objective Function (Average (std dev))	Objective Function (Best)	Total Time(s)	Objective Function	Time(s)	GAP
sd1	232.38(2.83)	<b>228.28</b>	1.876	228.28	0.7	<b>0.00%</b>
sd2	762.83(5.96)	760	2.76	<b>714.4</b>	54.4	6.38%
sd3	466.56(4.86)	458.25	2.985	<b>430.61</b>	67.3	6.42%
sd4	677.05(2.65)	676.28	3.019	<b>631.06</b>	400	7.17%
sd5	1520.91(13.68)	1484.85	4.898	<b>1408.12</b>	402.7	5.45%
sd6	860.44(0)	860.44	4.609	<b>831.21</b>	408.3	3.52%
sd7	3640(0)	<b>3640</b>	<b>6.154</b>	3714.4	403.2	<b>-2.00%</b>
sd8	5213.19(62.73)	<b>5106.5</b>	<b>8.204</b>	5200	404.1	<b>-1.80%</b>
sd9	2254.75(25.08)	2206.02	8.806	<b>2059.84</b>	404.3	7.10%
sd10	2853.12(36.29)	2757.51	12.588	<b>2749.11</b>	400	0.31%
sd11	13320(28.28)	<b>13280</b>	<b>19.278</b>	13612.12	400.1	<b>-2.44%</b>
sd12	7676.31(31.68)	7627.82	24.835	<b>7399.06</b>	408.3	3.09%
sd13	10559.42(44.6)	10470.09	28.642	<b>10367.06</b>	404.5	0.99%
sd14	11399.11(32.14)	11359.9	13.56	<b>11023</b>	5021.7	3.06%
sd15	15766.5(56.75)	15681.02	24.3	<b>15271.77</b>	5042.3	2.68%
sd16	3397.48(4.34)	<b>3391.7</b>	<b>18.18</b>	3449.05	5014.7	<b>-1.66%</b>
sd17	27532.4(83.43)	27407.36	31.05	<b>26665.76</b>	5023.6	2.78%
sd18	15007.04(77.58)	14853.66	31.227	<b>14546.58</b>	5028.6	2.11%
sd19	20635.12(172.20)	<b>20260.55</b>	<b>49.54</b>	20559.21	5034.2	<b>-1.45%</b>
sd20	41151.15(134.84)	40866.09	89.348	40408.22	5053	1.13%
sd21	11465.5(32.77)	<b>11389.72</b>	<b>474.05</b>	11491.67	5051	<b>-0.89%</b>

The GAP column in Table 3.2 and Table 3.3 is the percentage difference in objective function values of the hybrid GA and those obtained from Jin et al.(2008d) and Chen et al. (2007c) respectively. From Table 3.2, the hybrid GA was able to find solutions within 8%-17% for all the datasets. However, much greater success was found in improving the best known solutions in the 21 datasets of Chen et al. (2007c) .From Table 3.3, the hybrid GA found better solutions for 6 of the 21 datasets (sd7, sd8, sd11, sd16, sd19 and sd21) and were on par with the objective solution for one dataset (sd1) For the remaining datasets, the hybrid GA found solutions that were between 0.3% to 7.2% of the objective function but the computational times for hybrid GA were much faster for all the 21 datasets.

The objective function values for the two datasets are compared with the dual bound obtain by column generation (working paper, Wilck and Cavalier, 2012a), results of which are shown in Table 3.4 and Table 3.5 respectively. The GAP represents the percentage difference between the objective function values of ACO and the column generation dual bound. As you can see from Table 3.4 and Table 3.5 below, the percentage difference between ACO objective function and column generation dual bound ranges from 0 % to 6.7 % (2007c) and 11.80% to 18.56%(2008d) respectively.

**Table 3.4: Comparison of ACO objective function for Chen et al. (2007c) and Column generation dual bound (Working paper, Wilck and Cavalier)**

<b>Dataset</b>	<b>ACO Objective function</b>	<b>Column generation dual bound*</b>	<b>GAP</b>
sd1	228.28	228.28	0.00%
sd2	760	708.28	6.81%
sd3	458.25	430.58	6.04%
sd4	676.28	631.05	6.69%
sd5	1484.85	1390.57	6.35%
sd6	860.44	831.21	3.40%
sd7	3640	3640.00	0.00%
sd8	5106.5	5068.28	0.75%
sd9	2206.02	2044.23	7.33%
sd10	2757.51	2684.84	2.64%
sd11	13280	13265.29	0.11%
sd12	7627.82	7275.97	4.61%
sd13	10470.09	10093.72	3.59%
sd14	11359.9	10632.67	6.40%
sd15	15681.02	15146.92	3.41%
sd16	3391.7	3375.95	0.46%
sd17	27407.36	25320.09	7.62%
sd18	14853.66	14253.94	4.04%
sd19	20260.55	19768.23	2.43%
sd20	40866.09	38071.58	6.84%
sd21	11389.72	11062.32	2.87%

*\*Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.*

*Column Generation stopping criteria: 5% GAP [i.e.,  $GAP = (Primal\ Solution - Dual\ Bound) / Primal\ Solution$ ].*

**Table 3.5: Comparison of ACO objective function for Jin et al. (2008d) and Column generation dual bound (Working paper, Wilck and Cavalier)**

Dataset	ACO Objective function	Column generation dual bound*	GAP
s51d2	845.86	688.83	18.56%
s51d3	1080.32	920.58	14.79%
s51d4	1752.79	1520.71	13.24%
s51d5	1512.46	1310.12	13.38%
s51d6	2398.47	2115.20	11.81%
576d2	1282.8	1093.39	14.77%
s76d3	1649.51	1399.37	15.16%
s76d4	2357.02	2039.11	13.49%
s101d2	1586.97	1395.25	12.08%
s101d3	2122.04	1859.36	12.38%
s101d5	3109.88	2704.63	13.03%

*\*Column Generation cpu specifications: CPLEX and FORTRAN 95, GNU, Intel Xeon, 2.49 GHz, 8 GB RAM.*

*Column Generation stopping criteria: 5% GAP [i.e.,  $GAP = (Primal\ Solution - Dual\ Bound) / Primal\ Solution$ ].*

## 6. Conclusions and Future directions

This paper focused on solving instances of SDVRP from previous literature using a hybrid GA that consists of ACO, GA, and a heuristics to build route for SDVRP. Based on the results from Table 3.2 and Table 3.3, the hybrid GA were able to provide better results for the datasets from Chen et al. (2007c) and at a faster computational time as compared to the datasets from Jin et al. (2008d). I speculate that the nature of the datasets in Jin et al. (2008d) may be the reason for such results (i.e., these data sets were random; whereas the other data sets had patterns). One of the route improvement strategies is to have a candidate list to determine the next location for each customer in which only a set of predetermined closest locations are included in the candidate list. In previous research Bullnheimer et al. (1999a), irrespective of the problem size, the size of the candidate list was set to one fourth of the total number of customers. Hence, in future, I would like to incorporate a candidate list in our hybrid GA. Also, in future, I would like to test the hybrid GA on other variants of vehicle routing problem.

## 7. References

- Aleman, R.E., Zhang, X., & Hill, R. R. (2010d). An adaptive memory algorithm for the split delivery vehicle routing problem. *Journal of Heuristics*, 16(3), 441-473.
- Ambrosino, D., & Sciomachen, A. (2007a). A food distribution network problem: a case study. *IMA J. Manag. Math.*, 18(1), 33-53.
- Archetti, C., Sperenza, M.G., & Hertz, A. (2006b). A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem. *Transportation Science*, 40(1), 64-73.
- Archetti, C., Savelsbergh, M. W. P., & Speranza, M. G. (2008a). To split or not to split: That is the question. *Transportation Research, Part E* 44(1), 114-123.
- Archetti, C., Speranza, M. G., & Savelsbergh, M.W.P. (2008b). An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem. *Transportation Science*, 42(1),22-31.
- Bullnheimer, B., Hartl, R.F. & Struss, C. (1999a). An improved ant system algorithm for the vehicle routing problem. . *Annals of Operations Research*, 89, 319-328.
- Boudia, M., Prins, C., & Reghioiui, M. (2007b). *An effective memetic algorithm with population management for the split delivery vehicle routing problem*. Paper presented at the Proceedings of the 4th International Conference on Hybrid Metaheuristics, Dortmund, Germany, 16-30.
- Chen, S., Golden, B., & Wasil, E. (2007c). The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4), 318-327.
- Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002b). A guide to vehicle routing heuristics. *The Journal of the Operational Research Society*, 53(5), 512-522.
- Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, 6(1), 80-91.
- Dror, M., & Trudeau, P. (1989a). Savings by split delivery routing. *Transportation Science*, 23, 141-145.
- Dror, M., Laporte, G. & Trudeau, P. (1994a). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3), 239-254.

- Dror, M., & Trudeau, P. (1990). Split Delivery Routing. *Naval Research Logistics* 37, 383-402.
- Frizzell, P., & Giffin, J. (1992b). The bounded split delivery vehicle routing problem with grid network distances. *Asia-Pacific Journal of Operational Research*, 9, 101-116.
- Goldberg, D.E. (1989b). *Genetic Algorithms in Search, Optimization and Machine Learning*. reading, Addison-Wesley Longman Publishing Co., Inc, Boston, MA.
- Jin, M., Liua, K., & Eksioglu, B. (2008d). A column generation approach for the split delivery vehicle routing problem. *Operations research letters*, 36(2), 265-270.
- Mota, E., Campos, V., Corberán, Á., & Van Hemert, J. (2007d). A New Metaheuristic for the Vehicle Routing Problem with Split Demands., *Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science*, 4446, 121-129.
- Mullaseril, P. A., Dror, M., & Leung, J. (1997b). Split-delivery routing heuristics in livestock feed distribution. *J. Oper.Res. Soc.* , 48(2), 107-116.
- Nakao, Y., & Nagamochi, H. (2007e). A DP-based Heuristic Algorithm for the Discrete Split Delivery Vehicle Routing Problem. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, 1(2), 217-226.
- Sierksma, G., & Tijssen, G.A. (1998c). Routing helicopters for crew exchanges on off-shores locations. *Annals of Operations. Research*, 76, 261-286.
- Tavakkoli-Moghaddam, R., Safaei, N., Kah, M.M.O, & Rabbani, M. (2007f). A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. *Journal of the Franklin Institute, Modeling, Simulation and Applied Optimization Part II* , 344(5), 406-425.
- Wilck, J.H. & Cavalier, T.M. (2012a). A Column Generation Method for the Split Delivery Vehicle Routing Problem using a Route-Based Formulation, Working Paper, University of Tennessee, Knoxville and Pennsylvania State University.

**CHAPTER IV**  
**A GENETIC ALGORITHM APPROACH TO SOLVE THE**  
**PHYSICIAN SCHEDULING PROBLEM**

## **Publication Statement**

This is a joint work between Gautham P. Rajappa, Dr. Joseph H. Wilck, and Dr. Charles Noon. We are working on this paper for publication.

## **Abstract**

Emergency departments have repeating 24-hour cycles of non-stationary Poisson arrivals and high levels of service time variation. The problem is to find a shift schedule that considers queuing effects and minimizes average patient waiting time and maximizes physicians' shift preference subject to constraints on shift start times, shift durations and total physician hours available per day. An approach that utilizes a genetic algorithm and discrete event simulation to solve the physician scheduling problem in a hospital is proposed. The approach is tested on real world datasets for physician schedules.

## **1. Introduction**

Over the past two decades, genetic algorithms are being applied in solving complex real world combinatorial optimization problems such as vehicle routing, sequencing and scheduling of jobs on single machines and multiple machines, knapsack and bin packing problems, resource scheduling, and inventory problems. According to Fukunaga et al. (2002a) , a staff scheduling problem is known to be an NP-complete problem. Hence, metaheuristics such as genetic algorithms and Tabu Search are a commonly used methodology to solve such problems.

Every hospital faces a challenge of preparing a staff schedule based on the availability and preferences of the staff. A good work schedule should not only reduce the labor cost but also allow for more opportunities and a high degree of satisfaction amongst the staff. In addition, the staffs have to be scheduled in such a way that there are minimal or considerable waiting times for patients. Hence, the research objective of this chapter is to utilize a genetic algorithm to build physician shift schedules based on constraints such as physicians' preferences, their working hours and average patient wait times. The approach is tested on real-world datasets for physician schedules.

The rest of the chapter is organized as follows: Section 2 focuses on the literature associated with staff scheduling, Section 3 explains the problem and genetic algorithm approach in detail, and Results, Conclusions and future research are described in Section 4.

## **2. Literature Review**

According to Fukunaga et al. (2002a) a staff scheduling problem is known to be an NP-complete problem. Hence, one of the ways to obtain a feasible set of solutions in a reasonable amount of time frame is by application of heuristic and metaheuristics methods. Dean (2008a) proposed a two genetic algorithm (heuristic) solutions that applies a bit-string and a two dimensional chromosome structure for staff scheduling. In particular, Dean (2008a) modeled a staff schedule in the form of a two dimensional chromosome structure, in which the rows and columns represented the employees and days respectively. He compared these results to the results obtained by a bit-string structure (chromosomes) representation of a staff schedule. Downsland (1998a) proposed a Tabu Search and strategic oscillation approach to schedule the nurse roster in a major UK hospital. Easton and Mansour (1999) proposed a distributed genetic algorithm to tackle problems related to generalized set covering (GSC), deterministic goal programs (DGP), and stochastic goal programs (SGP). The distributed genetic algorithm used penalty functions for infeasible offspring and also employed a local search algorithm to enhance the performance. The DGP was tested on three different sets of data and it provided better solutions but at the expense of computational time.

Aickelin and Downsland (2004) developed an indirect approach in which initially a heuristic decoder builds the staff schedule from various combinations of available resources. Then a genetic algorithm was applied to optimize the output schedule from the heuristic decoder. The genetic algorithm only solved an unconstrained problem leaving the constraint handling to the heuristic decoder that uses them to directly bias the search rather than in penalty functions alone. Also, all problem specific knowledge was held in the heuristic decoder, thus enabling the algorithm to quickly adapt to changes in

problem specifications. The results obtained by this indirect approach were found to be more favorable and robust than those obtained by a Tabu search approach. Tanomaru (1995b) used genetic algorithm to solve staff scheduling problem with no predefined shift intervals. Hence instead of having predefined shift intervals, the planning horizon was split into uniform time intervals and staffs were assigned accordingly. Also, after every iteration, a number of heuristics were applied to improve the solution. Results were found to be optimal for small instances and good for large instances of the problem.

Jan et al. (2000b) used genetic algorithms to schedule nurses in a hospital using the concept of hard and soft constraints. The objective was to minimize the penalty function for violating the soft constraints and reduce the variance in individual nurse schedule to ensure fairness of schedule. Jan et al. (2000b) also suggested a method to allow the decision maker to adjust a schedule and direct the search during its execution.

Cai and Li (2000a) presented a genetic algorithm to solve the nurse scheduling problem with the following three objectives in decreasing order of importance: (1) Minimize total cost, (2) Minimize staff surplus, and (3) Minimize the variance in staff surplus. Predefined weekly schedules were assigned when the optimal number of workers for each schedule is found. Heuristics were then applied to resolve the constraints that were violated. The results were of good quality and were incorporated into the existing scheduling system.

Puente et al. (2009b) proposed a combination of heuristic decoder and genetic algorithm approach to schedule doctors in an emergency department. They used the concept of hard and soft constraints wherein weights were assigned to the soft constraints based on their importance. Actual results obtained by using this heuristic method have achieved a more balanced shift-assigning among the doctors with a high degree of satisfaction. Ohki et al. (2008b) developed a cooperative genetic algorithm (CGA) which uses crossover operator and periodically, the mutation operator to solve the nurse scheduling problem. They used penalty functions for evaluating the difference of the part of the shift schedule

between the original schedule given at the beginning of the current month and the schedule to be newly optimized.

To tackle the scheduling problem in a Belgian hospital, Burke et al. (1998b) developed a commercial heuristic solution called Plane in which the heuristic was a combination of Tabu search and algorithms based on manual scheduling techniques. Plane can decide (per nurse) which duties can or cannot be performed (according to that nurse's qualification category) when there is not enough personnel available and also provides an objective schedule in which all nurses are treated equally and the number of violated constraints is relatively low.

Inoue et al. (2003c) proposed an interactive scheduling approach wherein the fitness function was based on a measure of violation of soft constraints. However, at each iteration of solution generation, the users were given the opportunity to modify the schedule based on their opinion. The genetic algorithm used combinations of crossover, mutation and heuristics for repairing the crossover (new generation). Brusco and Jacobs (1993) proposed simulated annealing approach to address the cyclic staff scheduling problem. Their heuristic provided high quality solutions in a short computational time on a test dataset. They also suggested that branch-and-bound integer programming was impractical to solve cyclic staff scheduling problems.

Burke et al. (2009a) proposed a scatter search algorithm to schedule nurses in a hospital. In contrast to heuristics which work with one set of solutions, a scatter search algorithm works with a population of solutions. A scatter search algorithm is similar to memetic algorithms except that the random decisions are replaced with intelligently designed rules and solutions created from more than one parent. The results of the scatter search algorithm with hill climbing improvement method were found to be more optimal when tested against benchmark problems. Burke et al. (2001) used memetic algorithms for nurse scheduling and concluded that although memetic algorithm produces highly quality solutions, it requires a greater computation time than tabu search. Özcan (2005)

developed a memetic approach to solve a nurse rostering problem wherein the planning horizon was two weeks of shift schedule. Özcan (2005) used the hill climbing method to evaluate and repair each constraint that violated the shift schedule. In order to minimize the total staff with different experience levels subject to several labor agreements, Brunner and Edenharter (2011) formulated a staff scheduling problem as mixed integer linear program and solved it using a column generation based heuristics at the anesthesia department of a hospital.

Dias et al. (2003b) developed a tabu search and a genetic algorithm for solving the rostering problem in Brazilian hospitals wherein the soft constraints were weighted based on their priority and was used in the objective function. Results on test dataset showed that the genetic algorithm slightly outperformed Tabu search but, in practice, both approaches were well received by the hospital staff. A wide variety of numerous other operations research methods like column generation, constraint programming, Pareto optimization, mixed integer programming, hyperheuristics etc. have been applied to solve the staff scheduling problem, overviews of which can be found in the survey papers by Ernst et al. (2004a).

Paul et al. (2010) presented a systematic review of emergency department simulation literature from 1970 to 2006. Jacobson et al. (2006a) conducted a survey on various discrete event simulation models relevant to hospitals. Also, Jun et al. (1999a) have conducted an extensive survey on application of discrete event simulation in healthcare. Kumar and Kapur (1989a) used simulation to analyze alternatives to schedule nurses in emergency room at Georgetown University Hospital. Rosetti et al. (1999c) applied simulation to test various alternatives of emergency department physicians staffing schedules and to analyze the impact of the schedules on patient throughput and resource utilization. Weng et al. (2012) proposed a bi-level framework called multi-tool integrated methodology (MTIM) to schedule staff for each emergency room across various hospitals (distributed resource allocation decision) within the budget limitations.

Gendreau et al. (2007a) proposed four different scheduling techniques namely: tabu search, constraint programming, mathematical programming and column generation to schedule physicians in emergency department at five different hospitals in Canada. Yeh and Lin (2007b) proposed a combination for simulation modeling and genetic algorithms to improve quality of care in emergency department. The simulation model was used for analysis of flow of patients in the emergency department and genetic algorithm was used to develop a nurse schedule with the objective of minimizing patient wait time. Laskowski et al. (2009c) applied agent based models and queuing models to evaluate patient access and patient flow through emergency department. Xiao et al.(2010a) proposed a time window based incremental resource scheduling methodology (dynamic scheduling) that uses a genetic algorithm to schedule and reschedule resources based at selected points(time windows). To study the effectiveness of their methodology, their approach was integrated with an existing discrete event simulation system.

Though not in healthcare industry, Pantel et al. (1998c) applied a two step approach that had a combination of genetic algorithm and discrete event simulation for solving job shop scheduling problems in a semiconductor industry. In the first step, they used discrete event simulation to model the dynamic system behavior and in the second step, they applied genetic algorithm to minimize the average residence time to produce a set of batches in function of batch order in a multipurpose-multiobjective plant with unlimited storage. The discrete event simulation model was embedded in the optimization loop to evaluate the objective function. In our approach to solve the physician scheduling problem in healthcare, we also embed our discrete event solution module into the genetic algorithm, details of which are explained in Section 3.

### **3. Problem Definition and Genetic Algorithm approach**

#### ***3.1 Problem Definition***

In a typical emergency room at a hospital, patients arrive at random times and these arrival rates vary with respect to time of the day. Also, the services of the physicians are stochastic in nature. Constraints such as physicians preferences on shift start time and

shift duration, average patient waiting times and restriction on total working hours for all the physicians per day makes it a very complicated problem to solve. Hence, an efficient staff schedule algorithm should consider all these real world constraints and produce a result which satisfies both the physicians as well as the patients. A genetic algorithm approach is proposed in this paper to solve the staff scheduling problem and is tested on two datasets.

### 3.1.1 Datasets

The given data for the two datasets is shown in Table 4.1. For the two datasets, the average number of patients arriving per hour is assumed to be Poisson arrivals and the service times are assumed to be exponential distributed.

**Table 4.1: Given Data**

<b>Given Data</b>	<b>Dataset 1</b>	<b>Dataset2</b>
<b>Average service time</b>	15 minutes (exponential distribution)	33 minutes (exponential distribution)
<b>Average number of patients arriving per hour</b>	Poisson Arrivals (Table 4.2)	Poisson Arrivals (Table 4.3)
<b>Maximum physician hours per day</b>	48	68
<b>Feasible shifts with preference</b>	Table 4.4	Table 4.5

**Table 4.2: Average number of patients arriving per hour (Dataset 1)**

<b>Hour of the day</b>	<b>Average number of patients arriving</b>	<b>Hour of the day</b>	<b>Average number of patients arriving</b>
<b>12:00 AM</b>	3.690616	<b>12:00 PM</b>	8.178273
<b>1:00 AM</b>	2.911858	<b>1:00 PM</b>	7.79489
<b>2:00 AM</b>	2.293054	<b>2:00 PM</b>	7.792522
<b>3:00 AM</b>	2.017725	<b>3:00 PM</b>	8.053659
<b>4:00 AM</b>	1.831175	<b>4:00 PM</b>	7.983501
<b>5:00 AM</b>	1.856022	<b>5:00 PM</b>	7.969416
<b>6:00 AM</b>	2.251625	<b>6:00 PM</b>	8.282366
<b>7:00 AM</b>	3.803911	<b>7:00 PM</b>	7.664413
<b>8:00 AM</b>	5.446445	<b>8:00 PM</b>	7.238266
<b>9:00 AM</b>	7.066014	<b>9:00 PM</b>	6.578026
<b>10:00 AM</b>	7.939452	<b>10:00 PM</b>	5.526836
<b>11:00 AM</b>	8.49382	<b>11:00 PM</b>	4.336112

**Table 4.3: Average number of patients arriving per hour (Dataset 2)**

<b>Hour of the day</b>	<b>Average number of patients arriving</b>	<b>Hour of the day</b>	<b>Average number of patients arriving</b>
<b>12:00 AM</b>	2.621795	<b>12:00 PM</b>	7.083333
<b>1:00 AM</b>	1.916667	<b>1:00 PM</b>	6.826923
<b>2:00 AM</b>	1.448718	<b>2:00 PM</b>	6.557692
<b>3:00 AM</b>	1.294872	<b>3:00 PM</b>	6.570513
<b>4:00 AM</b>	1.403846	<b>4:00 PM</b>	6.076923
<b>5:00 AM</b>	1.378205	<b>5:00 PM</b>	6.512821
<b>6:00 AM</b>	1.839744	<b>6:00 PM</b>	6.730769
<b>7:00 AM</b>	2.858974	<b>7:00 PM</b>	6.750000
<b>8:00 AM</b>	4.288462	<b>8:00 PM</b>	6.064103
<b>9:00 AM</b>	5.769231	<b>9:00 PM</b>	5.384615
<b>10:00 AM</b>	6.769231	<b>10:00 PM</b>	4.339744
<b>11:00 AM</b>	7.038462	<b>11:00 PM</b>	3.147436

**Table 4.4: Feasible shifts with preference (Dataset 1)**

Hour of the day	Shift duration (hours)		
	8	10	12
7:00 AM	6	5	3
11:00 AM	6	3	4
3:00 PM	6	3	1
7:00 PM	4	4	2
11:00 PM	2	3	3

**Table 4.5: Feasible shifts with preference (Dataset 2)**

Hour of the day	Shift durations (hours)				
	8	9	10	11	12
7:00 AM	4	6	6	3	2
8:00 AM	4	6	6	3	2
9:00 AM	4	6	6	4	2
10:00 AM	4	6	6	4	2
11:00 AM	4	6	6	4	2
12:00 PM	5	5	5	3	2
1:00 PM	5	6	5	4	2
2:00 PM	5	6	5	4	2
3:00 PM	5	5	5	4	2
4:00 PM	6	5	5	3	2
5:00 PM	6	5	3	3	2
6:00 PM	3				
9:00 PM	2	3	3	3	
10:00 PM	3	6	6	4	2
11:00 PM	3	5	5	3	2

From Table 4.4, for dataset 1, the shift start times are at 7AM, 11AM, 3PM, 7PM and 11PM. All shifts must start only at these times. The shift duration for each of the shift start times is 8, 10 or 12 hours. The preferences for each combination of shift start time and shift duration are shown in Table 4.4. The preferences are numbered from 1 to 6, 6 being the most preferred start time and shift duration, and 1 being the least preferred. The interpretation of feasible shifts with preferences for dataset 2 (Table 4.5) is similar to that of dataset 1.

### **3.1.2 Objective Functions and Constraints**

Based on the given data, the objectives and constraints for the two datasets are described below.

#### **Objectives**

Based on the given data, the objective is to build a shift schedule that:

- 1) Maximize the preference of physicians.
- 2) Minimize the average waiting time for patients.

#### **Constraints**

- 1) There is no overtime i.e. the shift schedule should not exceed the maximum physician hours per day.
- 2) At least one physician is available every hour.
- 3) Shifts can start only at times shown in the preference matrix (Table 4.4 and Table 4.5).

Since it is a multiobjective optimization problem, weights (penalties) are assigned to each objective and weighted sum is used to calculate the objective function value. Noon et al.(2007) had a mathematical formulation for the given problem and this formulation has been modified to suit our problem definition. The mathematical formulation for the problem is described in Section 3.1.3.

### **3.1.3 Mathematical Formulation**

#### **Indexed Sets:**

$i$  = time period from 1 ...  $T$ , where  $T$  is 24 hours

$j$  = shift index from 1 ...  $J$ , where  $J$  is the total number of potential shifts

$T_t$  = Total simulation run time

$i, j$  are integers

#### **Parameters:**

$\mu$  = service rate (constant, exponential distribution)

$\lambda_i$  = arrival rate for time period  $i$

$d_j$  = duration of each shift  $j$

$P_j$  = Preference of shift  $j$

$H$  = maximum available server(physician) hours  
 $\omega_1$  = weight associated with average patient wait time  
 $\omega_2$  = weight associated with shift preference penalty  
 $b_{ij} = \begin{cases} 1, & \text{if shift } j \text{ is assigned to time } i \\ 0, & \text{otherwise} \end{cases}$

**Decision Variables:**

$s_i$  = number of servers (physicians) available at time period  $i$   
 $x_j$  = number of servers (physicians) in shift  $j$

**Accounting Variables (Calculated from decision variables and discrete event simulation):**

$w_i$  = total average patient wait time for time period  $i$

**Objective:**

$$Z = \text{Min}(\omega_1 \sum_i^{T_t} w_i + \omega_2 \sum_j^J (P_j * x_j)) \quad (4.1)$$

**Constraints:**

- 1) Total physician hours is  $\leq H$  (maximum physician hours/day)

$$\sum_j^J (d_j * x_j) \leq H \quad (4.2)$$

- 2) The number of physicians in each shift must be equal to number of physicians every hour

$$\sum_j^J (b_{ij}) * (x_j) = s_i \quad \forall i \quad (4.3)$$

- 3) At least one physician every hour

$$s_i \geq 1, \text{ integer } \forall i \quad (4.4)$$

- 4) Number of physicians in a given shift

$$x_j \geq 0, \text{ integer } \forall j \quad (4.5)$$

**Solve for:**

$w_i = f([\lambda_i, \mu, s_i])$ : (Average patient waiting time function from **discrete event simulation**). The discrete event simulation module is an integral part of the proposed Genetic Algorithm to evaluate average patient wait times for each feasible shift schedule.

### ***3.2 Genetic Algorithm Approach***

Genetic algorithms are population based search algorithms to solve combinatorial optimization problems. It was first proposed by John Holland (1989). In these algorithms the search space (population) of a problem is represented as a collection of individuals (chromosomes) and these individuals are evaluated based on the fitness function. Genetic algorithms generate solutions for optimization problem based on theory of evolution using concepts such as reproduction, crossover and mutation. The fundamental concept of a genetic algorithm states a set of conditions to achieve global optima. These conditions describe the reproduction process and ensure that better solution remain in future generations and weaker solutions be eliminated from future generations. This is similar to the Darwin's survival of fittest concept in the theory of evolution. A typical genetic algorithm consists of the following steps (1989):

- Step 1:** Generate an initial population of  $N$  solutions.
- Step 2:** Evaluate each solution of the initial population using a fitness function/objective function.
- Step 3:** Select solutions as parents for the new generation based on probability or randomness. The best solutions (in terms of fitness or objective) have a higher probability of being selected than poor solutions.
- Step 4:** Use the parent solutions from Step 3 to produce the next generation (called offspring). This process is called as crossover. The offspring are placed in the initial set of solutions replacing the weaker solutions.
- Step 5:** Randomly alter the new generation by mutation. Usually this is done using a mutation probability.
- Step 6:** Repeat Steps 2 through 5 until a stopping criteria is met.

Due to the constraints of this problem, it is not possible to directly use genetic algorithm in the way it is described above. In particular, after crossover, there may be solutions which do not satisfy the constraints. Hence, to obtain a feasible set of offspring, we may need to modify the way crossover is done or another possibility is to remove infeasible solutions after mutation and replace them with the solutions having higher fitness value

in the old population (2002b) or complete the new population with a schedule heuristics. In our approach, if an infeasible solution exists for future generation, we randomly select new shift schedules from the initial population. The genetic algorithm approach for dataset 1 is explained below.

### **Solution Encoding**

In Dataset 1, the queuing system is stable (calculated from given data) and a maximum of 48 physician hours is available per day. Hence, we simply make decisions on shifts by generating random shift schedule and evaluating its fitness function. The randomly generated shift schedules will define how many servers we have on at each hour. The fitness function will determine how well the capacity handled the demand or whether there would be large queues. We have three shift durations of 8, 10 or 12 hours. Hence, the maximum number of shift required would be simply the available number of physician hours (48 hours) divided by the least shift duration (i.e., 8 hours). Hence we require a maximum of 6 shifts.

As we have 15 preferences, each preference index in the preference matrix (Table 4.6) is numbered from 0 to 14 row wise. For example, index 0 is a 7AM shift with shift duration of 8 hours and index 14 is an 11PM shift with duration of 12 hours. A no schedule is assigned the number 15.

**Table 4.6: Shift index (Shift preference) matrix (Dataset 1)**

<b>Shift index (preferences)</b>	<b>Shift duration (hours)</b>		
	<b>8</b>	<b>10</b>	<b>12</b>
<b>Hour of the day</b>			
<b>7:00 AM</b>	0(6)	1(5)	2(3)
<b>11:00 AM</b>	3(6)	4(3)	5(4)
<b>3:00 PM</b>	6(6)	7(3)	8(1)
<b>7:00 PM</b>	9(4)	10(4)	11(2)
<b>11:00 PM</b>	12(2)	13(3)	14(3)
<b>No schedule</b>	15		

### **Step 1: Initial Population**

For the initial population, I first randomly generate 2000 shift schedules of size 6 (maximum number of shifts). For example, one shift schedule may be [0,3,4,5,14,15] and another shift sequence may be [4,15,4,9,10,11]. Then each of the 2000 randomly generated shift schedule is evaluated to verify if there is at least one physician available every hour and there is no overtime in the shift schedule (maximum of 48 physician hours per day). If a randomly generated shift schedule has at least one physician every hour and there is no overtime, this shift schedule is added to the initial population. This process continues until a predetermined number of initial population is generated which in our case is set to 500.

### **Step 2: Evaluation of the fitness function**

It involves two steps as shown below:

**1) Validity of the shift sequence:**

This is done to verify if there is at least one physician available every hour and there is no overtime in the shift schedule (maximum of 48 physician hours per day). If a randomly generated shift schedule has at least one physician every hour and there is no overtime, this shift schedule is added to the population.

**2) Evaluation of Fitness Function:**

For every shift schedule in the population, its fitness function is calculated based on 2 objectives 1) Maximize physician preference and 2) Minimize the average patient wait time.

- **Maximize physician preference:** A penalty of (6- preference for that particular shift) is imposed. For example, for a 7AM, 8 hour shift, the penalty is  $6 - 6 = 0$ .
- **Minimize the average patient wait time:** A 2400 hour (100 days \* 24 hours/day) discrete event simulation is implemented for each of the shift schedules based on patient arrival rate and availability of physicians per hour.

- Then the convex combination of weights (penalty) for each of the above two objectives is used to evaluate the fitness function.

### **Step 3: Selection**

The randomly generated shift schedules are sorted accordingly to the lowest fitness value. The top 100 shift sequences are then selected for future generation.

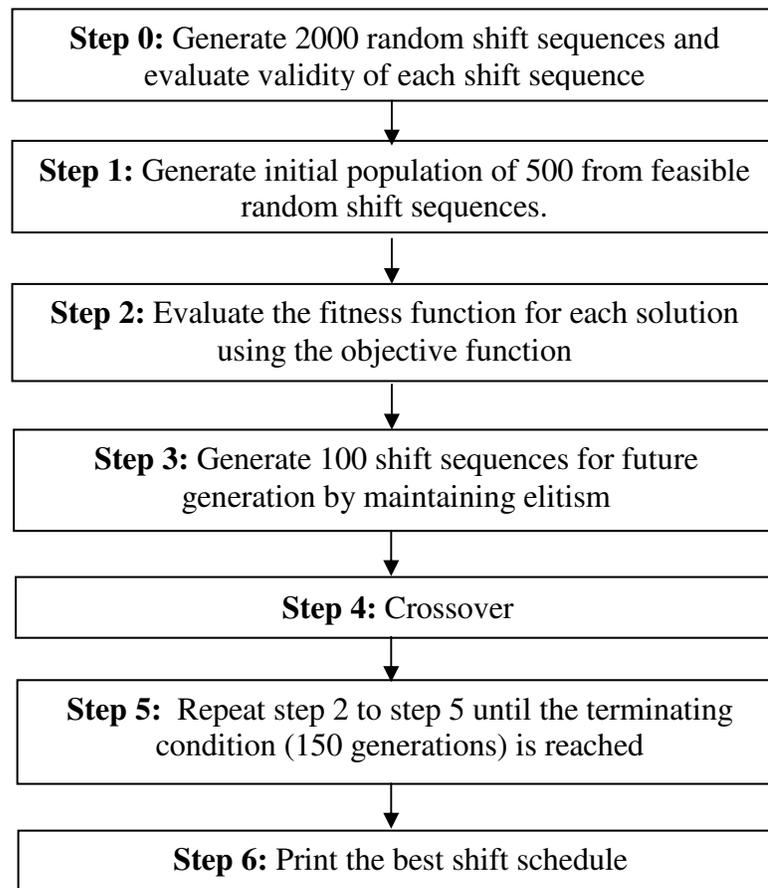
### **Step 4: Crossover**

- **Elitism:** The top 5 shift schedules from the selection step are always added to the future generation.
- **Parent Selection:** From the selection pool of shift schedules, 2 parents are randomly selected and two children of shift sequences are generated using one-point crossover for the new generation.
- The crossover probability is set to 1.
- There is no mutation.
- If feasible schedules cannot be found, I randomly add feasible schedules to the new generation until the population size of 100 is reached.

### **Step 5: Terminating condition**

Then step 2, 3 and 4 is repeated for a fixed number of generations (terminating condition), which in our problem is set to 150.

The genetic algorithm flowchart for dataset 1 is shown in Figure 4.1 below:



**Figure 4.1: Genetic Algorithm Flowchart (Dataset 1)**

For Dataset 2, the queuing system is stable (calculated from given data) and maximum of 68 physician hours is available per day. Hence, the maximum number of shift required would be simply the available physician hours (68 hours) divided by the least shift duration (i.e., 8 hours). Hence we require a maximum of 9 shifts. As we have 70 preferences wherein each preference index in the preference matrix (Table 4.7) is numbered from 0 to 69 row wise. A no schedule is assigned the index 70. Besides the solution encoding, the genetic algorithm approach for dataset 2 is similar to that of dataset 1. Due to the problem size, the genetic algorithm parameters such as population size, number of generations etc were increased by a factor of 3 for dataset 2 as compared to dataset 1. Also, please note that the genetic algorithm parameters such as population size, terminating condition etc. were all set during pilot- testing.

**Table 4.7: Shift index (Shift preference) matrix (Dataset 2)**

Shift index(preference) Hour of the day	Shift duration (hours)				
	8	9	10	11	12
7:00 AM	0(4)	1(6)	2(6)	3(3)	4(2)
8:00 AM	5(4)	6(6)	7(6)	8(3)	9(2)
9:00 AM	10(4)	11(6)	12(6)	13(4)	14(2)
10:00 AM	15(4)	16(6)	17(6)	18(4)	19(2)
11:00 AM	20(4)	21(6)	22(6)	23(4)	24(2)
12:00 PM	25(5)	26(5)	27(5)	28(3)	29(2)
1:00 PM	30(5)	31(6)	32(5)	33(4)	34(2)
2:00 PM	35(5)	36(6)	37(5)	38(4)	39(2)
3:00 PM	40(5)	41(5)	42(5)	43(4)	44(2)
4:00 PM	45(6)	46(5)	47(5)	48(3)	49(2)
5:00 PM	50(6)	51(5)	52(3)	53(3)	54(2)
6:00 PM	55(3)				
9:00 PM	56(2)	57(3)	58(3)	59(3)	
10:00 PM	60(3)	61(6)	62(6)	63(4)	64(2)
11:00 PM	65(3)	66(5)	67(5)	68(3)	69(2)
No schedule	70				

## 4. Results, Conclusions, and Future Work

### 4.1 Results

The genetic algorithm for this study was coded in Java on a Windows7, Intel i5 2.4 Ghz, 4 GB RAM computer. The discrete event simulation module to evaluate average patient wait time was also coded in Java and was integrated with the genetic algorithm to generate shift schedules. The algorithm was run for convex combination of weights for the objective functions. Due to its simplicity, a weighted sum approach was used to calculate the objective function (Abdullah et al. (2006)). The results for a convex combination of weights ranging from 0 to 1 for dataset 1 and dataset 2 are shown in Table 4.8 and Table 4.9 respectively.

**Table 4.8: Weighted sum approach results (Dataset 1)**

Case #	Preference Weight	Average patient wait time Weight	GA Time(sec)	Total Preference Violation	Average patient wait time(min)	Total physician hours	Shift Schedule
<b>1</b>	<b>1</b>	<b>0</b>	340.964	3	37.94	44	[0, 6, 14, 15, 0, 6]
<b>2</b>	<b>0.9</b>	<b>0.1</b>	116.315	3	20.89	44	[14, 0, 3, 6, 15, 6]
<b>3</b>	<b>0.8</b>	<b>0.2</b>	116.923	3	20.89	44	[6, 15, 14, 6, 3, 0]
<b>4</b>	<b>0.7</b>	<b>0.3</b>	118.778	4	15.31	48	[6, 0, 6, 0, 12, 3]
<b>5</b>	<b>0.6</b>	<b>0.4</b>	118.633	4	15.31	48	[12, 3, 6, 0, 0, 6]
<b>6</b>	<b>0.5</b>	<b>0.5</b>	118.827	4	15.31	48	[3, 12, 6, 0, 0, 6]
<b>7</b>	<b>0.4</b>	<b>0.6</b>	119.131	5	15.31	48	[14, 0, 5, 15, 3, 6]
<b>8</b>	<b>0.3</b>	<b>0.7</b>	118.623	4	15.31	48	[3, 6, 12, 6, 0, 0]
<b>9</b>	<b>0.2</b>	<b>0.8</b>	119.079	8	13.88	48	[11, 0, 1, 15, 4, 6]
<b>10</b>	<b>0.1</b>	<b>0.9</b>	119.995	8	13.88	48	[1, 15, 11, 0, 4, 6]
<b>11</b>	<b>0</b>	<b>1</b>	119.41	8	13.88	48	[6, 1, 11, 15, 4, 0]

**Table 4.9: Weighted sum approach results (Dataset 2)**

Case #	Preference Weight	Average patient wait time Weight	GA Time(sec)	Total Preference Violation	Average patient wait time (min)	Total Physician hours	Shift Schedule
1	1	0	1170.017	0	35.95	64	[21, 1, 70, 36, 45, 1, 22, 62, 70]
2	0.9	0.1	1101.017	0	35.68	65	[17, 2, 7, 45, 36, 70, 70, 61, 11]
3	0.8	0.2	1145.259	2	35.46	64	[45, 12, 70, 7, 30, 6, 22, 66, 70]
4	0.7	0.3	1092.167	2	34.39	66	[21, 2, 32, 70, 11, 61, 12, 70, 51]
5	0.6	0.4	1122.037	3	31.21	66	[1, 12, 70, 7, 30, 17, 42, 70, 66]
6	0.5	0.5	899.491	3	30.94	67	[2, 62, 1, 70, 70, 21, 32, 50, 13]
7	0.4	0.6	917.144	6	30.27	67	[21, 11, 70, 33, 36, 70, 68, 1, 26]
8	0.3	0.7	910.54	12	27.28	68	[69, 30, 2, 16, 70, 70, 24, 21, 55]
9	0.2	0.8	926.297	10	26.88	68	[68, 31, 27, 70, 46, 3, 0, 12, 70]
10	0.1	0.9	907.925	13	26.38	67	[45, 10, 19, 70, 70, 8, 27, 5, 67]
11	0	1	885.737	16	25.58	67	[5, 70, 16, 19, 68, 70, 23, 55, 5]

As you can see from Table 4.8 and Table 4.9 above, for dataset 1 and dataset 2, a zero weight to the average patient wait time objective function results in an average patient time of 37.94 minutes and 35.95 minutes respectively, and when no weight is assigned to preferences of the physicians, the average patient wait time is 13.88 minutes and 25.58 minutes respectively. Also, as the preference weight decreases from 1 to 0 and average patient wait time weight increases from 0 to 1, the total preference violation increases and the average patient wait time decreases for the two datasets. The computational time for the genetic algorithm is shown in the fourth column (GA Time (sec)). As you can see for dataset 1 in Table 4.8, for the first 3 cases, wherein the physician preference has more weight, the total physician hours used is only 44 hours as compared to the maximum of 48 hours available each day. Whereas for dataset 2 in Table 4.9, there are only two instances (case #8 and case #9) wherein the maximum available physician hours of 68 hours is completely used.

The shift schedules for each convex combination of weights are shown in the last column in Table 4.8 and Table 4.9. For example, for case #2 in dataset 1, the best shift schedule is [14,0,3,6,15,6]. Using Table 4.6, the shift schedule is as follows:

- **14** → Start shift at 11PM for 12 hours
- **0** → Start shift at 7AM for 8 hours
- **3** → Start shift at 11AM for 8 hours
- **6** → Start shift at 3PM for 8 hours
- **15** → No schedule
- **6** → Start shift at 3PM for 8 hours

A similar interpretation can be done for all the cases in the two datasets. The plot of total preference violation v/s. average patient wait time for all convex combinations of weight for dataset 1 and dataset 2 is shown in Figure 4.2 and Figure 4.3 respectively.

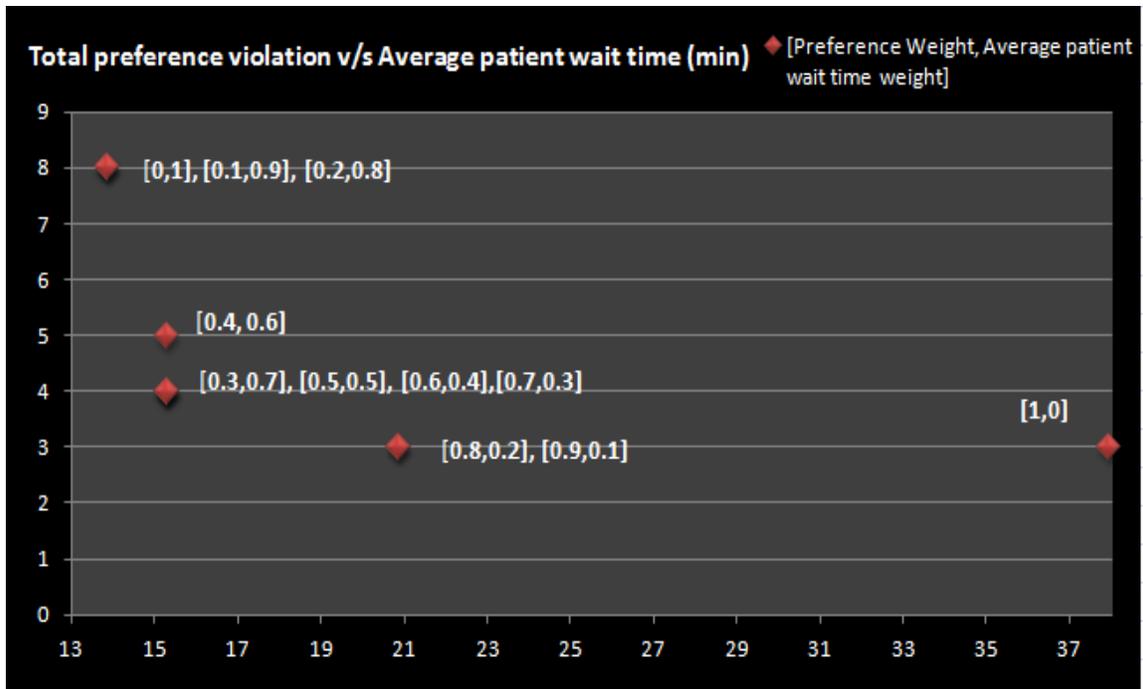


Figure 4.2: Total preference violation v/s Average patient wait time (min)(Dataset 1)

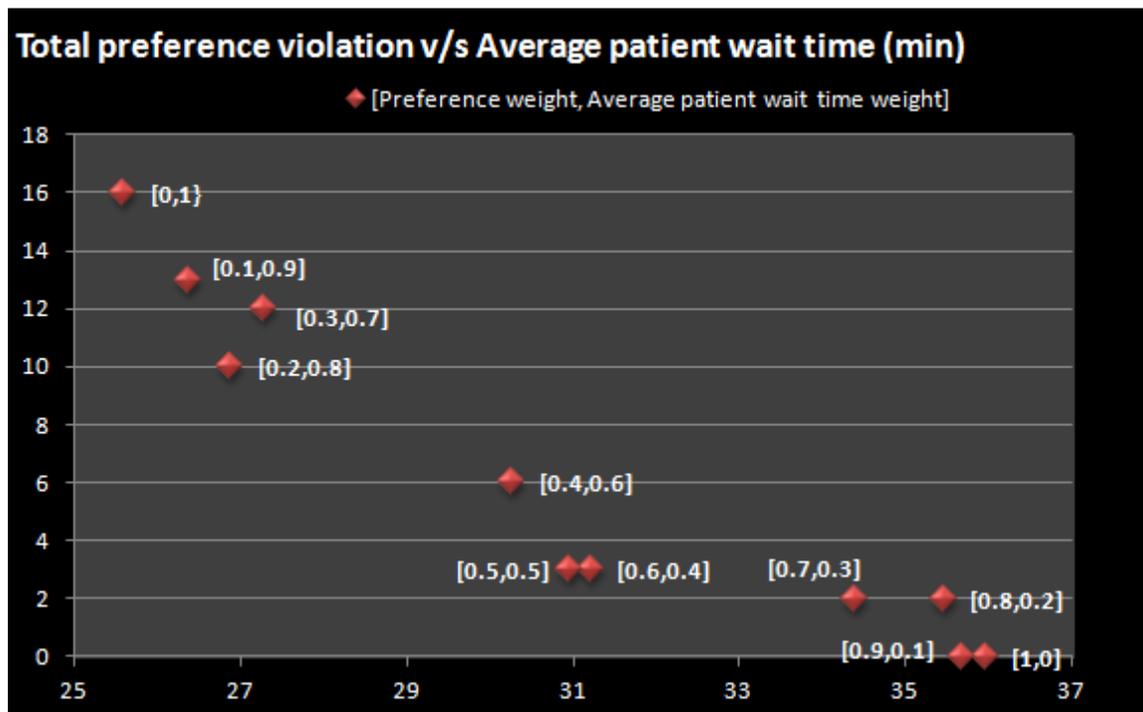


Figure 4.3: Total preference violation v/s Average patient wait time (min)(Dataset 2)

For Case #2, Case #6 and Case #11 in dataset 1, the number of doctors available per hour and the “number of patients of capacity” is shown in Table 4.10 and a plot showing how the shift schedule handles the patient arrivals each hour is shown in Figure 4.4(A), Figure 4.4(B) and Figure 4.4(C) respectively. The “number of patients of capacity” shows the amount of patients that can be served by physicians every hour for each shift schedule.

**Table 4.10: Number of patients of capacity (Dataset 1)**

Hour of the day	Average number of patient arrivals/hr	Case # 2		Case # 6		Case # 11	
		Available physicians/hr	Number of patients of capacity	Available physicians/hr	Number of patients of capacity	Available physicians/hr	Number of patients of capacity
12:00 AM	3.690616	1	4	1	4	1	4
1:00 AM	2.911858	1	4	1	4	1	4
2:00 AM	2.293054	1	4	1	4	1	4
3:00 AM	2.017725	1	4	1	4	1	4
4:00 AM	1.831175	1	4	1	4	1	4
5:00 AM	1.856022	1	4	1	4	1	4
6:00 AM	2.251625	1	4	1	4	1	4
7:00 AM	3.803911	2	8	2	8	2	8
8:00 AM	5.446445	2	8	2	8	2	8
9:00 AM	7.066014	2	8	2	8	2	8
10:00 AM	7.939452	2	8	2	8	2	8
11:00 AM	8.493820	2	8	3	12	3	12
12:00 PM	8.178273	2	8	3	12	3	12
1:00 PM	7.794890	2	8	3	12	3	12
2:00 PM	7.792522	2	8	3	12	3	12
3:00 PM	8.053659	3	12	3	12	3	12
4:00 PM	7.983501	3	12	3	12	3	12
5:00 PM	7.969416	3	12	3	12	2	8
6:00 PM	8.282366	3	12	3	12	2	8
7:00 PM	7.664413	2	8	2	8	3	12
8:00 PM	7.238266	2	8	2	8	3	12
9:00 PM	6.578026	2	8	2	8	2	8
10:00 PM	5.526836	2	8	2	8	2	8
11:00 PM	4.336112	1	4	1	4	1	4

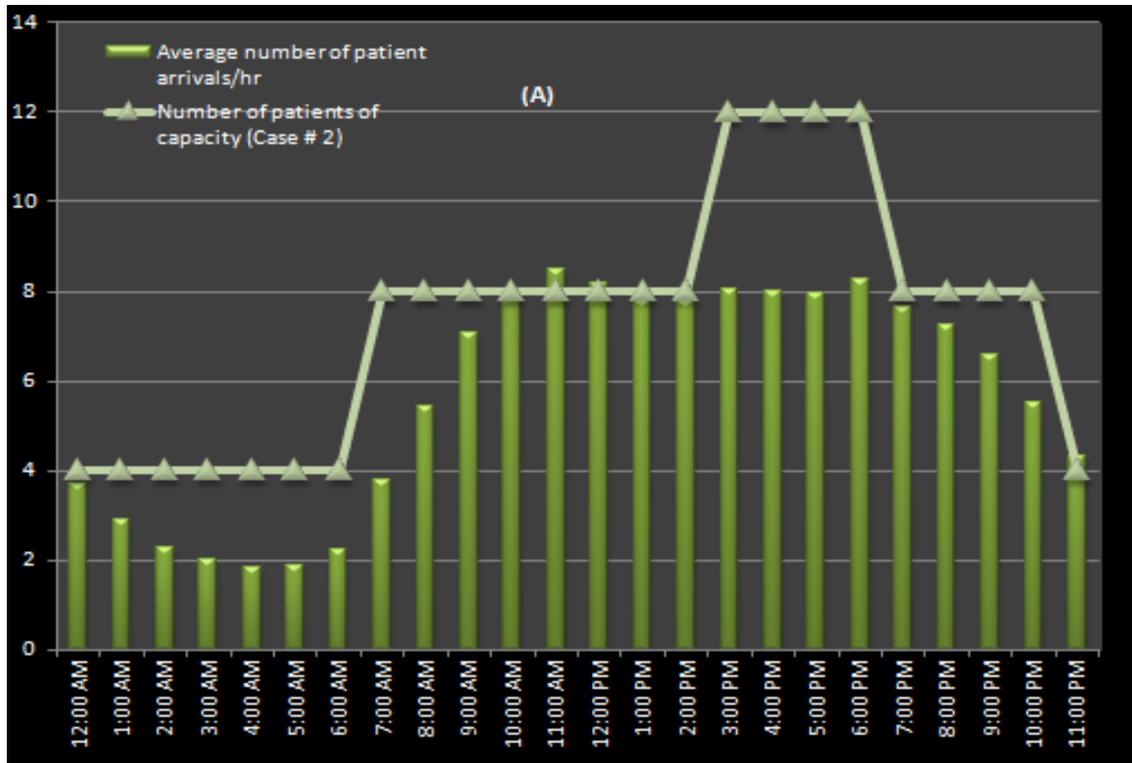


Figure 4.4(A): Number of patients of capacity plot (Case # 2, Dataset 1)

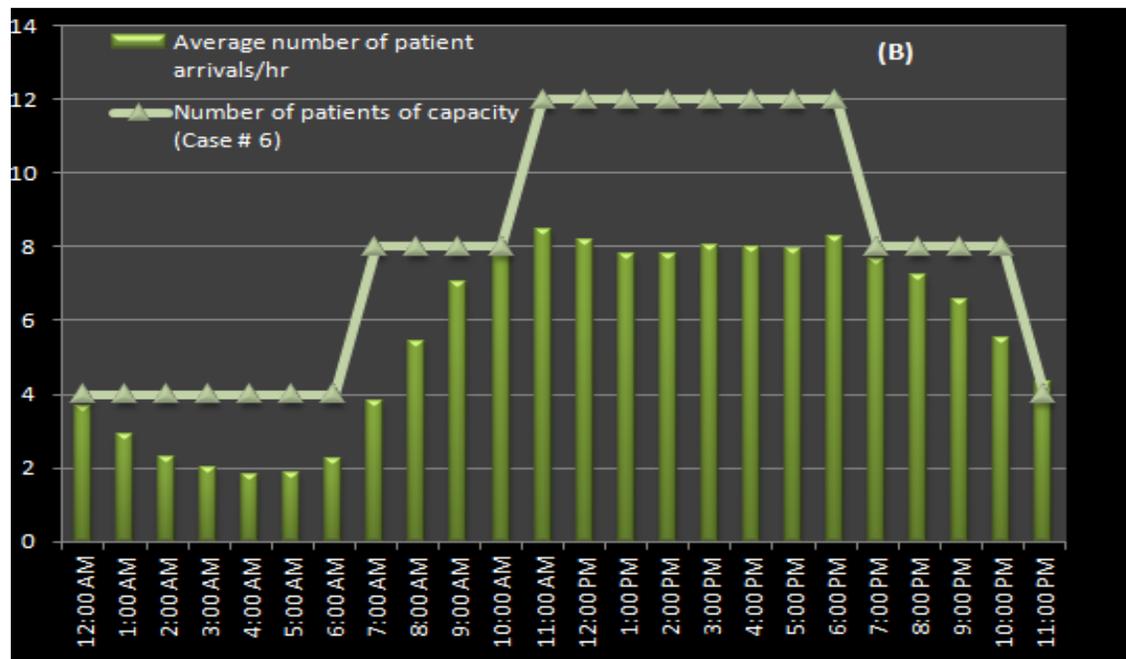
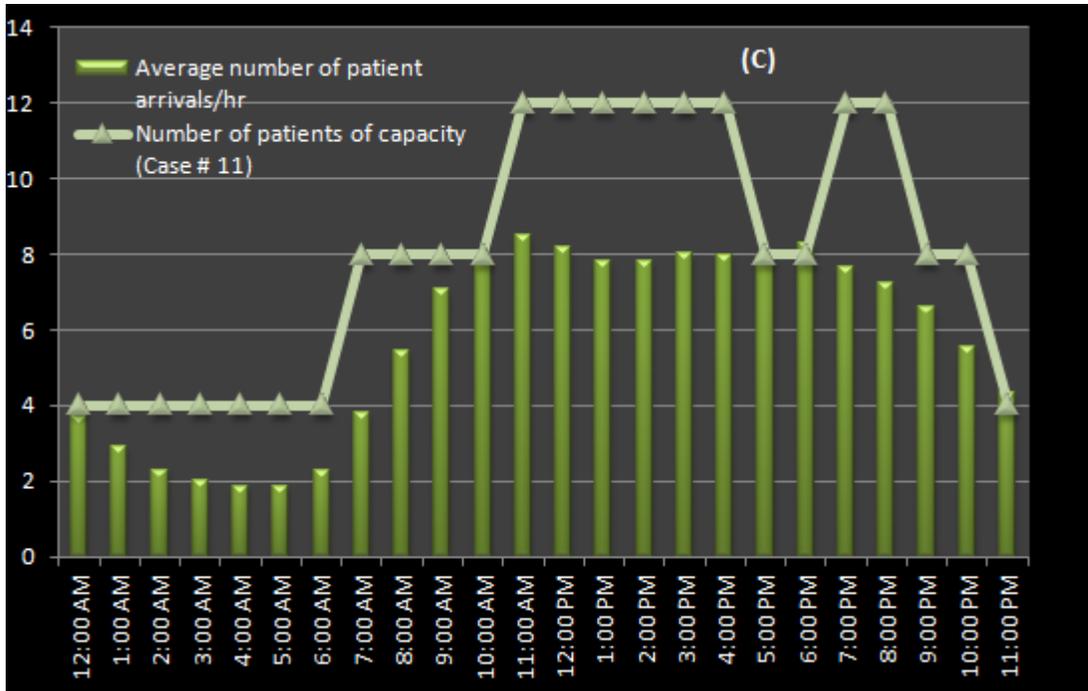


Figure 4.4(B): Number of patients of capacity plot (Case # 6, Dataset 1)



**Figure 4.4(C): Number of patients of capacity plot (Case # 11, Dataset 1)**

The columns in the plots above represent average patient arrival rate for every hour and the lines represent the physicians’ capacity to serve the patients. As you can see from the three plots above, when the weights are more towards reducing the patient average wait time as compared to physicians’ preference (Figure 4.4(C)), the genetic algorithm generates shift schedules that tend to add capacity during peak patient arrival hours as compared to Case # 2 , wherein the physicians’ preference have more weight. Hence, the addition of extra capacity results in less patient average wait time (Case # 11) as compared to Case # 2.

Similarly, for Case #2, Case #6 and Case #11 in dataset 2, the number of doctors available per hour and the “number of patients of capacity” is shown in Table 4.11 and a plot showing how the shift schedule handles the patient arrivals each hour in shown in Figure 4.5(A), Figure 4.5(B) and Figure 4.5(C) respectively. The plots for these cases can be interpreted in the same manner in which they were interpreted for Dataset 1.

**Table 4.11: Number of patients of capacity (Dataset 2)**

Hour of the day	Average number of patient arrivals/hr	Case # 2		Case # 6		Case # 11	
		Available physicians/hr	Number of patients of capacity	Available physicians/hr	Number of patients of capacity	Available physicians/hr	Number of patients of capacity
12:00 AM	2.621795	1	1.82	2	3.64	2	3.64
1:00 AM	1.916667	1	1.82	1	1.82	2	3.64
2:00 AM	1.448718	1	1.82	1	1.82	1	1.82
3:00 AM	1.294872	1	1.82	1	1.82	1	1.82
4:00 AM	1.403846	1	1.82	1	1.82	1	1.82
5:00 AM	1.378205	1	1.82	1	1.82	1	1.82
6:00 AM	1.839744	1	1.82	1	1.82	1	1.82
7:00 AM	2.858974	1	1.82	3	5.46	1	1.82
8:00 AM	4.288462	2	3.64	2	3.64	3	5.46
9:00 AM	5.769231	3	5.45	3	5.46	3	5.46
10:00 AM	6.769231	4	7.27	3	5.46	4	7.28
11:00 AM	7.038462	4	7.27	4	7.28	5	9.1
12:00 PM	7.083333	4	7.27	4	7.28	5	9.1
1:00 PM	6.826923	4	7.27	5	9.10	5	9.1
2:00 PM	6.557692	5	9.09	5	9.10	5	9.1
3:00 PM	6.570513	5	9.09	5	9.10	5	9.1
4:00 PM	6.076923	6	10.91	4	7.28	3	5.46
5:00 PM	6.512821	5	9.09	4	7.28	3	5.46
6:00 PM	6.730769	3	5.45	4	7.28	4	7.28
7:00 PM	6.750000	3	5.45	4	7.28	3	5.46
8:00 PM	6.064103	2	3.64	2	3.64	3	5.46
9:00 PM	5.384615	2	3.64	2	3.64	3	5.46
10:00 PM	4.339744	3	5.45	3	5.46	1	1.82
11:00 PM	3.147436	2	3.64	2	3.64	2	3.64

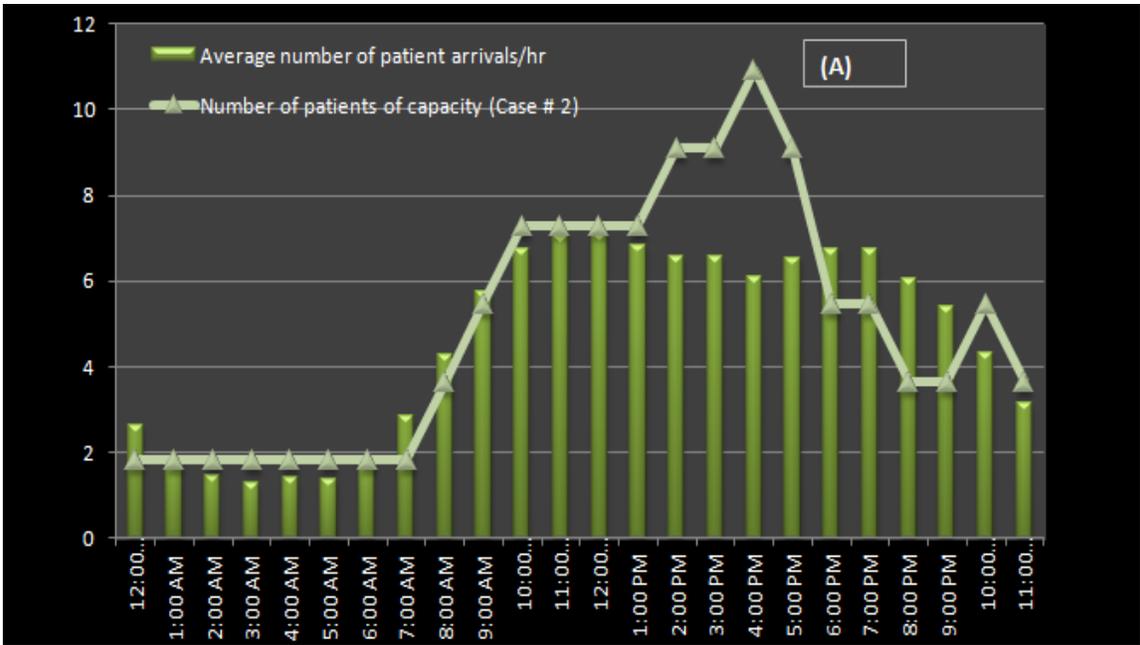


Figure 4.5(A): Number of patients of capacity plot (Case # 2, Dataset 2)

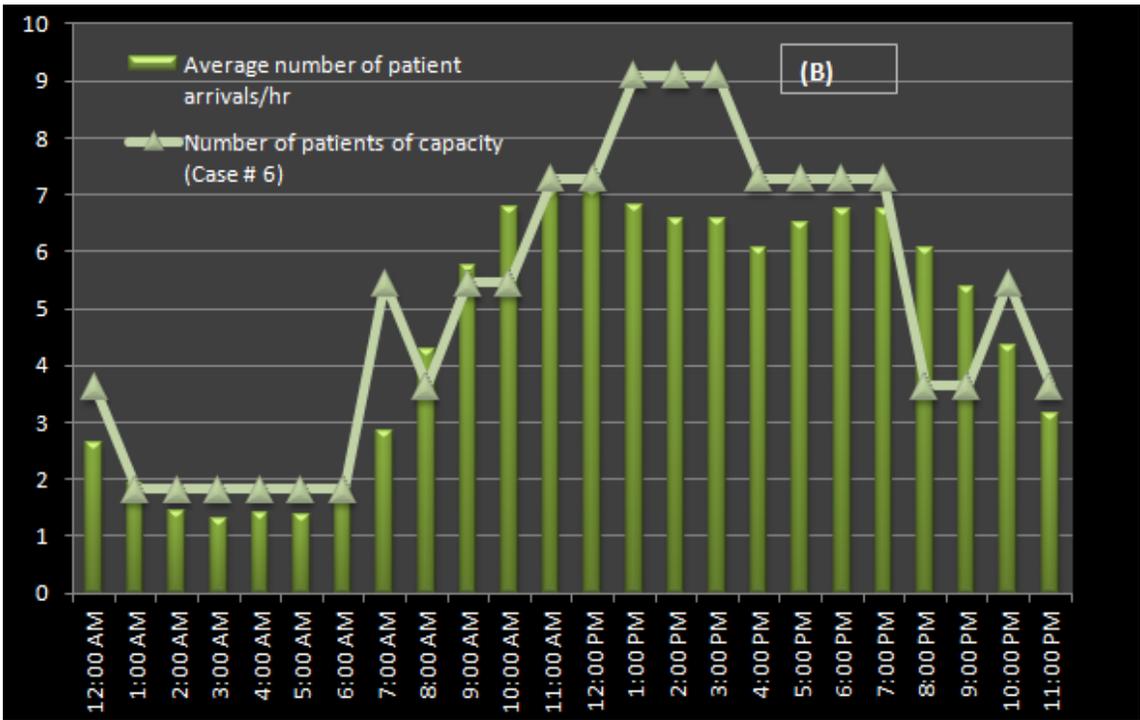
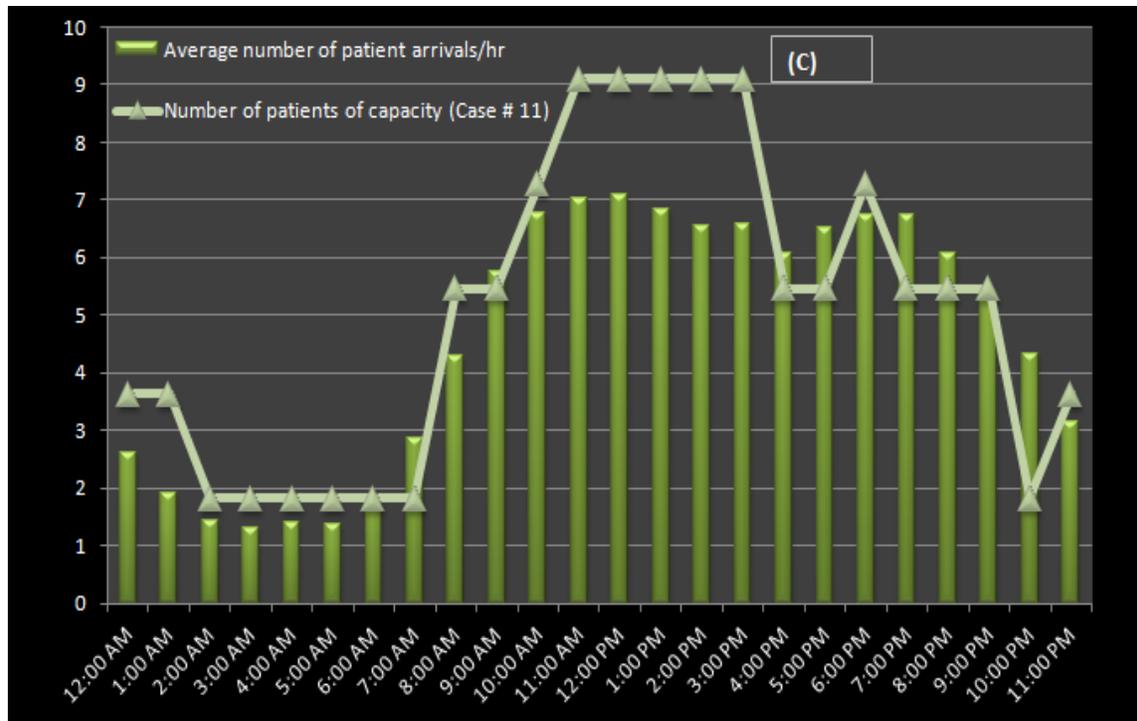


Figure 4.5(B): Number of patients of capacity plot (Case # 6, Dataset 2)



**Figure 4.5(C): Number of patients of capacity plot (Case # 11, Dataset 2)**

#### **4.2 Conclusions and Future Work**

This paper provides a genetic algorithm approach to solve the staff scheduling problem. As noted by Michalewicz (1995a), the results of a genetic algorithm are very problem specific and the proposed genetic algorithm is also very specific to the problem. Also, discrete event simulation was embedded in the genetic algorithm to evaluate the patient average wait time. One of the main drawbacks of using weighted sum approach is that the objective function is very sensitive to weights. Hence, in future, I would like to use an alternate approach proposed by Hajela and Lin (1992), in which multiple solutions can be obtained in a single run. Also, this problem only considers an overall physician schedule. In future, I would like to modify my genetic algorithm in such a way that it can generate schedules for every individual physician.

### **5. References**

- Aickelin, U., & Dowsland, K. (2004). An Indirect Genetic Algorithm for a Nurse Scheduling Problem. *Computers & Operations Research*, 31(5),761-778.
- Azzaro-Pantel, C., Bernal-Haro, L., Baudet, P., Domenech, S., & Pibouleau, L.,(1998c).

- A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm, *Computers & Chemical Engineering*, 22(10), 1461-1481.
- Brunner, J., & Edenharter, G.(2011), Long term staff scheduling of physicians with different experience levels in hospitals using column generation, *Health Care Management Science*, 14(2), 189-202.
- Brusco, M., & Jacobs, L. (1993). A simulated annealing approach to the cyclic staff-scheduling problem. *Naval Research Logistics*, 40(1), 69-84.
- Burke, E.K., Cowling, P., De Causmaecker, P.D., & Berghe, G. V. (2001). A Memetic Approach to the Nurse Rostering Problem. *Applied Intelligence*, 15(3), 199-214.
- Burke, E.K., Causmaecker, P.D, & Berghe, G. V. (1998b). *A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem*. Paper presented at the Second Asia-Pacific Conference on Simulated Evolution and Learning,187-194.
- Burke, E.K., Curtois, T., Qu, R. & Berghe, G. V. (2009a). A Scatter Search Methodology For The Nurse Rostering Problem. *Journal of the Operational Research Society*, 61, 1667-1679.
- Cai, X., & Li, K. N. (2000a). A genetic algorithm for scheduling staff of mixed skills under multi-criteria. *European Journal of Operational Research*, 125, 359-369.
- Cordeau, J. F., Gendreau, M., Laporte, G., & Potvin, J.-Y. (2002b). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53, 512-522.
- Dean, J. S. (2008a). *Staff Scheduling by a Genetic Algorithm with a Two-Dimensional Chromosome Structure*. Paper presented at the 7th Conference on the Practice and Theory of Automated Timetabling, Montreal,Canada.
- Dias, T. M., Ferber, D.F., Souza, C.C., & Moura, A. V. (2003b). Constructing nurse schedules at large hospitals. *International Transactions in Operational Research*, 10(3), 245-265.
- Downsland, K. A. (1998a). Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research*, 393-407.

- Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., & Sier, D. (2004a). An Annotated Bibliography of Personnel Scheduling and Rostering. *Annals of Operations Research*, 127, 21-144.
- Fukunaga, A.E., Fama, J., Andre, D., Matan, O., & Nourbakhsh, I. (2002a). *Staff Scheduling for Inbound Call Centers and Customer Contact Centers*. Paper presented at the Eighteenth National Conference on Artificial intelligence, 822-829.
- Gendreau, M. , Ferland, J. , Gendron, B. , Hail, N. , Jaumard, B. , Lapierre, S. , Pesant, S. , & Soriano, P. (2007a). *Physician scheduling in emergency rooms*, Proceedings of the 6th international conference on Practice and theory of automated timetabling VI, , Brno, Czech Republic, 53-67.
- Goldberg. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.*, MA: Addison-Wesley.
- Hajela, P. & Lin, Y. (1992). Genetic search strategies in multicriterion optimal design *Structural Optimization*, 4(2), 99-107.
- Inoue, T., Furuhashi, T., Maeda, H., & Takaba, M. (2003c). A Proposal of Combined Method of Evolutionary Algorithm and Heuristics for Nurse Scheduling Support System. *IEEE Transactions on Industrial Electronics*, 50(5), 833-838.
- Jacobson, S.H., Hall, S.N., Swisher, J.R. (2006a). *Discrete-Event Simulation of Health Care Systems*, International Series in Operations Research & Management Science, 91, 211-252.
- Jan, A., Yamamoto, M., & Ohuchi, A. (2000b). *Evolutionary Algorithms for Nurse Scheduling Problem*. Paper presented at the Proceedings of the 2000 Congress on Evolutionary Computation. , San Diego, California, USA,196-203.
- Jun, J.B., Jacobson, S.H., Swisher, J.R. (1999a). Application of discrete-event simulation in health care clinics: a survey. *J Oper Res Soc* 50, 109–123
- Konak, A., Coit, D.W., & Smith, A.E., (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91, 992–1007.
- Kumar, A., & Kapur, A. (1989a). *Discrete Simulation Application - Scheduling Staff for*

- the Emergency Room*, Winter Simulation Conference Proceedings, ed. E.A. MacNair, K.J. Musselman, and P. Heidelberger, IEEE, Washington, DC, 1112-1120.
- Laskowski, M., McLeod, R.D., Friesen, M.R., Podaima, B.W., & Alfa, A.S. (2009c) Models of Emergency Departments for Reducing Patient Waiting Times. *PLoS ONE*,4(7).
- Mansour, N., & Easton, F. F. (1999). A distributed genetic algorithm for deterministic and stochastic labor scheduling problems. *European Journal of Operational Research*, 118, 505-523.
- Michalewicz, Z. (1995a). *A Survey of Constraint Handling Techniques in Evolutionary Computation Methods*. Paper presented at the 4th Annual Conference on Evolutionary Programming,135-155.
- Noon, C., Clay, A., & Crane, J. (2007). Determining Shift Schedules for 24-hour Queuing Systems with Varying Arrival Rates.
- Ohki, M.,Uneme, S. & Kawano, H. (2008b). *Parallel Processing of Cooperative Genetic Algorithm for Nurse Scheduling*. Paper presented at the 4th International IEEE Conference "Intelligent Systems",2,10-36 – 10-41.
- Özcan, E. (2005). *Memetic Algorithms for Nurse Rostering*. Paper presented at the The 20th International Symposium on Computer and Information Sciences,482-492.
- Paul, S.A., Reddy, M.C., & Deflitch, C.J.(2010), A Systematic Review of Simulation Studies Investigating Emergency Department Overcrowding, *Simulation*, 86(8-9), 559-571.
- Puente, J., Gómez, A., Fernández, I. & Priore, P. (2009b). Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. *Computers & Industrial Engineering*, 56(4), 1232-1242.
- Rossetti, M.D., Trzcinski, G. F., & Syverud, S.A. (1999b). *Emergency department simulation and determination of optimal attending physician staffing schedules*, Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future, Phoenix, Arizona, United States, (2), 1532-1540

- Tanomaru, J. (1995b). *Staff scheduling by a genetic algorithm with heuristic operators*. Paper presented at the Proceedings of the IEEE Conference on Evolutionary Computation, 1,456.
- Weng,S., Wu, T., Mackulak,G.T., & William A. Verdin.(2012). A multi-tool integrated methodology for distributed resource allocation in healthcare, *International Journal of Industrial and Systems Engineering*,11(4),428-452.
- Xiao, Junchao and Osterweil, Leon J. and Wang, Qing.(2010a). *Dynamic scheduling of emergency department resources*, Proceedings of the 1st ACM International Health Informatics Symposium, Arlington, VA, USA,590-599.
- Yeh, J., & Lin, W.(2007b). Using simulation technique and genetic algorithm to improve the quality care of a hospital emergency department, *Expert Systems with Applications*, 32(4), 1073-1083.

**CHAPTER V**  
**CONCLUSION**

## **1. Chapter Abstract**

In this dissertation, genetic algorithm and ant colony optimization was applied to solve combinatorial optimization problems in the field of logistics and healthcare staff scheduling. In particular, two chapters focus on solving SDVRP using genetic algorithms and ant colony optimization. Another chapter applied genetic algorithm to solve a real world emergency department staff scheduling problem.

## **2. Chapter Highlights**

The highlights of each chapter are as follows:

### **Chapter 2: Ant Colony Optimization for the Split Delivery Vehicle Routing Problem**

- For the first time ever, Ant Colony Optimization was applied to the Split Delivery Vehicle Routing Problem.
- The ACO algorithm found competitive solutions for two benchmark problem sets.
- In some instances, ACO found the best ever solution for the test problem.
- Candidate list size plays a key role in the first ever application of ACO to SDVRP.

### **Chapter 3: A hybrid Genetic Algorithm approach to solve the Split Delivery vehicle routing problem**

- A hybrid genetic algorithm consisting of genetic algorithm, heuristics and ant colony optimization was developed to solve the SDVRP.
- The hybrid genetic algorithm found competitive solutions for two benchmark problem sets.

### **Chapter 4: A Genetic Algorithm approach to solve the physician scheduling problem**

- A genetic algorithm was developed to solve a real world physician schedule problem.

- The problem was a multi objective optimization problem wherein the physicians' shifts were scheduled based on their preferences of shift start time and duration ,no overtime and in patients' point of view, reduce their average wait time.
- The average wait time for patients were calculated using a discrete event simulation module and was part of the genetic algorithm.

### **3. Future Directions**

The GA and ACO work shown in this dissertation for the SDVRP could be applied to other VRP variants with some modification to account for additional constraints, likewise additional study of the candidate list issues could be explored. Finally, using GA and ACO in conjunction with an exact method (e.g., column generation) could be explored to find both an integer feasible solution and a dual solution (to raise the lower bound) in order to solve to optimality.

The GA procedure for the physician scheduling was specific to that problem; however, it could be extended to schedule multiple physicians across multiple facilities (e.g., hospital systems with more than one site). It could also be used in conjunction with scheduling other resources (e.g., nurses and physicians), where the decisions is further convoluted by having nurse and provider schedules that are dependent.

## **VITA**

Gautham P. Rajappa was born in Puttur, Karnataka, India. He completed his high school from K.V.NAL, Bangalore in 2000. In 2004, he got his Bachelor of Engineering (B.E.) degree in Mechanical Engineering from the National Institute of Engineering (NIE), Mysore, India (affiliated to VTU, Belgaum, India). He then got his M.S. from University of Wisconsin-Madison in December 2007 and Ph.D. from University of Tennessee, Knoxville in August 2012.