SIMULATED ANNEALING ALGORITHM FOR TRAVELLING SALESMAN PROBLEM

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

in

ELECTRONICS AND COMPUTER ENGINEERING

(With Specialization in Control and Guidance)

By

AVJIT KUMAR GUPTA

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)

JUNE, 2010

CANDIDATE'S DECLARATION

hereby declare that the work, which is presented in this dissertation report, titled ***Simulated Annealing Algorithm for Travelling Salesman Problem"***, being ubmitted in partial fulfillment of the requirements for the award of the degree of **laster of Technology** with specialization in **Control and Guidance**, in the)epartment of Electronics and Computer Engineering, Indian Institute of 'echnology, Roorkee is an authentic record of my own work carried out from July 009 to June 2010, under guidance and supervision of **Dr. R. MITRA**, Professor,)epartment of Electronics and Computer Engineering, Indian Institute of 'echnology, Roorkee.

The results embodied in this dissertation have not submitted for the award of any other)egree or Diploma.

)ate : 28/06/2010

Avjit Kumar Gupta

Place: Roorkee

AVJIT KUMAR GUPTA

CERTIFICATE

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

(Dr. R. MITRA)
Professor, E&C Department,
Indian Institute of Technology, Roorkee
Roorkee – 247 667, (INDIA)

Acknowledgement

With great sense of pleasure and privilege, I take this opportunity to express my deepest sense of gratitude towards my supervisor and guide Dr. **R. Mitra** for his valuable suggestions, sagacious guidance, and scholarly advice to improve the quality of present work. His professionalism, suggestions and his ways of thinking inspired me, and this inspiration guides me at every point of my life. I consider myself extremely fortunate for having got the opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to all faculty members of **Control & Guidance** for their constant encouragement, caring words, constructive criticism and suggestions towards the successful completion of this work. My sincere thanks to laboratory staff for letting me access the computers and other resources at will, for completion of this work.

I would like to thank Mona Subramaniama and Mrs. Manju A. for their support in modeling the system and the valuable suggestions that led me towards right path. I am very thankful to Atul, Sharan, Rajesh, Vaibhav, Parag, Ajit, Sadanand, Gulshan, Aryan, Ankit, Mrigank, Vivek for providing their moral support and good company every time I needed, which always acted as a strength for me at every point.

Most of all I am highly indebted to my parents ,uncle, aunty, sisters, brother-in law, Prashant, Arushi, Samarth and all other family members and all other friends whose sincere prayers, best wishes, moral support and encouragement was a constant source of assurance, guidance, strength, and inspiration to me. Finally, I would like to extend my gratitude to all those persons who directly or indirectly contributed towards this work.

Abstract

Sensitivity and Robustness is the key factor while designing the controller for nonlinear systems. One of the performance objectives for controller design is to keep the error between the controlled output and the set-point as small as possible. The control of many non-linear, inherently unstable systems using conventional methods is both difficult to design and marginally satisfactory in implementation. The introduction of optimization techniques in control engineering that makes use of evolutionary computation and an implicit imprecision is successful in counteracting these limitations. The field of computational intelligence has incorporated to such systems with an objective to achieve higher optimality and satisfactory performance.

The main objective of this work is to design "Simulated Annealing Algorithm" for the well known "Travelling Salesman Problem". In "Travelling Salesman Problem" the primary goal is to reduce the cost (i.e. the distance travelled by the person to cover all the cities once and once and return to their starting city). Basically, the aim is to achieve "Global Minima" in the total search space given. In this algorithm, firstly a random route is selected and its cost is calculated then onwards, according to "Metropolis Criteria" acceptance and rejection of route are done to improve the cost.

For this purpose; the supporting theory of "Simulated Annealing Algorithm" and "Travelling Salesman Problem" were discussed. After that, the mathematical analysis for the convergence of the "Simulated Annealing Algorithm" was discussed. Then onward, the algorithm is implemented for a small size "Travelling Salesman Problem" (i.e. 10 cities "Travelling Salesman Problem"). Then, the size of the problem is increased in the small steps. After that a comparative study of the "Simulated Annealing Algorithm" And "Tabu Search Algorithm" has been done.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

LIST OF FIGURES vii

CHAPTER 1: Introduction 1

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Travelling Salesman Problem and Combinatorial Optimization 3

1.4 Methodology 3

1.5 Simulated Annealing 8

1.6 Overview of the Thesis 9

CHAPTER 2: Classical Simulated Annealing Algorithm 10

2.1 Introduction 10

2.2 Local Optimization 11

2.3 Statistical Mechanics- A Physical Analogy 12

2.4 Classical Simulated Annealing 18

CHAPTER 3: Quantitative Analysis of Simulated Annealing Algorithm 22

3.1 Introduction 22

3.2 Mathematical Model 24

3.2.1 Asymptotic Convergence 25

3.2.2 Annealing Schedule 29

3.3 Analysis of Cost Function 31

CHAPTER 4: Application of Simulated Annealing Algorithm 35

4.1 Inverted Pendulum 35

4.2 Ball and Beam System 37

4.2.1 Mechanical Model of Ball and Beam System 38

4.3 Magnetic Levitation System 41

CHAPTER 5: Simulation Results and Their Interpretation 44

CHAPTER 6: Conclusion and Future Prospective **58**

6.1 Conclusion 58

6.1 Future Prospective 59

References 60

LIST OF FIGURES

Figure No.	Title	Page No.
2.1	Local Optimization Algorithm	12
2.2	Plateau Local Minima and Global Minimum for the Cost Function	13
2.3	Boltzmann's distribution curve for an energy function at various Temperatures	15
2.4	General Metropolis Algorithm	16
2.5	Analogy between Physical System and Combinatorial Optimization	18
2.6	Simulated Annealing Algorithm	20
4.1	Structure of control Strategy for Ball & Beam system	37
4.2	Mechanical system of Ball and Beam	38
4.3	Flowchart of Inherited control algorithm for Ball and Beam System	40
5.1	Convergence Figure obtained by "Simulated Annealing Algorithm" for 10 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	44
5.2	Route obtained by "Simulated Annealing Algorithm" for 10 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	45
5.3	Convergence Figure obtained by "Simulated Annealing Algorithm" for 20 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And	45

	on Y "Cost" and Cooling Rate is 0.97)	
5.4	Route obtained by "Simulated Annealing Algorithm" for 20 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	46
5.5	Convergence Figure obtained by "Simulated Annealing Algorithm" for 30 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	46
5.6	Route obtained by "Simulated Annealing Algorithm" for 30 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	47
5.7	Convergence Figure obtained by "Simulated Annealing Algorithm" for 50 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	47
5.8	Route obtained by "Simulated Annealing Algorithm" for 50 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	48
5.9	Convergence Figure obtained by "Simulated Annealing Algorithm" for 75 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	48
5.10	Route obtained by "Simulated Annealing Algorithm" for 75 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	49

5.11	Convergence Figure obtained by "Simulated Annealing Algorithm" for 442 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	49
5.12	Route obtained by "Simulated Annealing Algorithm" for 442 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	50
5.13	Convergence Figure obtained by "Simulated Annealing Algorithm" for 535 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)	50
5.14	Route obtained by "Simulated Annealing Algorithm" for 535 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)	51
5.15	Convergence Figure obtained by "Tabu Search Algorithm" for 30 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")	51
5.16	Route obtained by "Tabu Search Algorithm" for 30 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)	52
5.17	Convergence Figure obtained by "Tabu Search Algorithm" for 50 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")	52
5.18	Route obtained by "Tabu Search Algorithm" for 50 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)	53
5.19	Convergence Figure obtained by "Tabu Search Algorithm" for 75 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y	53

	"Cost")	
5.20	Route obtained by "Tabu Search Algorithm" for 75 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)	54
5.21	Convergence Figure obtained by "Tabu Search Algorithm" for 442 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")	54
5.22	Route obtained by "Tabu Search Algorithm" for 442 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)	55
5.23	Convergence Figure obtained by "Tabu Search Algorithm" for 535 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")	55
5.24	Route obtained by "Tabu Search Algorithm" for 535 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)	56

CHAPTER 1

INTRODUCTION

1.1. Motivation:

The problem based on vehicle routing basically involves finding a set of delivery routes from one or several central depots to various demand points (e.g. consumers), in order to optimize a kind of objective function (minimization of routing costs, or the sum of fixed and variable costs, or the number of vehicles required, or the time consumed in following the route, etc.). There are some constraints on the vehicle like maximum rout time constraints, capacity constraints. For example, the problem arises when there is only a single vehicle of unlimited capacity, unit demands, only the routing costs, and an objective function which minimizes total distance travelled, is the famous Travelling Salesman Problem (TSP)[1].

Instead of minimizing the distance there may be several other type of objective function notations such as time, cost, and number of vehicle required in the fleet which may be considered equivalently. With several vehicles of common capacity, a single depot, known demands, and same objective function as Travelling Salesman Problem (TSP), there is a standard vehicle routing problem.

A huge amount of literature devoted only to Travelling Salesman Problem has made a good impact. One has to simply consult [1] to be convinced that Travelling Salesman Problem is one of the most fundamental and prominent, the most intensively investigated among all unsolved classical combinatorial optimization problems. Although everybody can easily explain and clearly conceptualize the Travelling Salesman Problem. It is, in fact the most difficult and first combinatorial type of problem.

The work done on this problem is partially reflection of the fact that the Travelling Salesman Problem encountered and represents a huge set of different kind of practical problems [1]. A specific and representative example of such practical problem is the application of Travelling Salesman Problem in the drilling holes on printed circuit board. The Travelling Salesman

Problem is the embedded component of most of the vehicle routing problem such as retail distribution, mail and newspaper delivery, municipal waste collection, fuel oil delivery etc. It also has some surprising application such as in robotics.

Motivated by extensive usability and applicability of "Travelling Salesman Problem" and its intensive computation, the most important objective of this thesis devoted toward developing a method to solve "Travelling Salesman Problem". The secondary objective of this thesis is to analyze the performance of this method with respect to other existing method for solving Travelling Salesman Problem. The third objective of this thesis is the computational study of Travelling Salesman Problem via examination of the quality of its final solution.

1.2. The Problem Statement:

The Travelling Salesman Problem is stated as follows:

Given a set of N cities coordinate by which we calculate the distance between any two cities, and a salesman is required to visit each and every city once and only once, starting from any city and returning to the city of departure; then "what is the shortest route, or tour, that he must choose in order to minimize the total distance he travelled?" with the assumption that there exist a direct straight path between any two cities. Instead of minimizing the distance there may be several other type of objective function notations such as time, cost, and number of vehicle required in the fleet which may be considered equivalently. Mathematically, it can be formulated as follows:

Given a set of cities vertex $V = \{v_0, ..., v_N\}$ and the distance, d_{ij}, from v_i to v_j, what is the best order cycle permutation, $\sigma = (\sigma(1), ..., \sigma(N))$ of V, that minimizes the cost function,

$$C(\sigma) = \sum_{i=1}^{N-1} d_{[\sigma(i),\sigma(i+1)]} + d_{[\sigma(N),\sigma(1)]} \qquad (1.1)$$

It is necessary to note that this permutation is selected by all cyclic permutations; there are $\frac{|(N-1)}{2}$ such permutations, in the Travelling Salesman Problem with symmetric distance matrix.

1.3. Travelling Salesman Problem and Combinatorial Optimization:

Combinatorial optimization is a subject which consists of problems that are central to many disciplines of science and technology engineering. Ongoing research in these areas has devoted towards developing the efficient methodologies and techniques for finding (minimum or maximum) the optimum values of a function of many independent variables. The function which we have to optimize is, usually called cost function, objective function, represent a measure of quantitative "goodness" of some complex systems. It is one of the important reason that the last three generation of combinatorial mathematician and operations research analysts, including computer scientist and engineers along with optimality control engineers, cumulatively have devoted literally many man years to study combinatorial optimization and extensive work on such problems.

Travelling Salesman Problem is one of the basic and most representative problem of all type of combinatorial optimization problem in general, it is an embedded component of most of the vehicle routing problems. It seems to be extremely interesting that how often the old method for solving Travelling Salesman Problem have precipitated and generated new and general techniques and methods in combinatorial optimization. The objective of this section is to introduce the reader with the flavor and a good appreciation of its historical importance and to highlight some key events.

From the earliest studies of discrete models, the Travelling Salesman Problem has been a major stimulant to research on combinatorial optimization. The early studies of the Travelling Salesman Problem pioneered the use of cutting plane techniques in integer programming and were responsible for several important ideas associated with tree enumeration method including coining the term "branch and bound". At that time, in the context of dynamic programming, problem of partitioning and decomposition introduced that will later proved to be fruitful in other applications of dynamics programming, and in accessing heuristic method of combinatorial optimization. An isolated probabilistic study of the Travelling Salesman Problem in the plane has become widely recognized as the seminal contribution to the probabilistic evaluation of heuristic methods for combinatorial optimization.

There are so many contributions to combinatorial optimization throughout 1950's and 1960's, for such problem classes as machine scheduling and production planning with setup cost, crew scheduling, set covering, and facility location problem were extensions and generalizations of these basic themes. Ongoing research work focused on the designing of optimization algorithms, usually based upon dynamic programming recursions or somewhat tailored versions of general purpose integer programming methods, very often for special case problem in the problem class.

At the same time that these integer and dynamic programming methods were evolving, combinatorial optimization was emerging and flourishing as a discipline in applied mathematics, based, in the large part, on the widespread practical and combinatorial applications of network flow theory and its generalizations such as nonbipartite matching and matroid optimization. Indeed, it is easy to understand the importance of these landmark contributions in defining combinatorial optimization as we know it today.

Although researchers were designing and applying heuristic (approximate) algorithms during the 1950's and 1960's for example, exchange heuristics for both the Travelling Salesman Problem and facility location problem, optimization based methods remained at the forefront of the academic activity. The heuristic algorithms developed at this time may have been progenitors of the algorithms studied later, but their analysis was often of such a rudimentary nature that heuristic did not capture the imagination and full acceptance of the academic community of this era. Rather than statistical assessment or error bound analysis, limited empirical verification of heuristics ruled the 1960's.

Two developments in the 1970's, namely the computation complexity theory and the evolution of enhanced capabilities in mathematical programming, revitalized combinatorial optimization and precipitated a new focus in its research. The familiar computation complexity theory has shown that the Travelling Salesman Problem and nearly every other "difficult" combinatorial optimization problem, the so called NP-complete (nondeterministic polynomial time complete) class of problems, are all computationally equivalent; namely, each of those problems has eluded any algorithmic design guaranteed to be more efficient than tree enumeration, and if one problem could be solved by an algorithm that is polynomial in its problem size, then they all could. This revelation suggested that algorithm possibilities for optimization methods were limited, and motivated renewed interest to design and analyze effective heuristics. Again, the Travelling

Salesman Problem was at the forefront during this era. Worst case (i.e. performance guarantee) analysis, statistical analysis, and probabilistic analysis of various heuristics for the Travelling Salesman Problem typified this period of research and were among the first steps in the evolution of interesting analytical approaches for evaluating heuristic methods. Indeed, the mere fact that computational complexity theory embraced the "infamous" Travelling Salesman Problem undoubtedly was instrumental in the theory's acceptance as a new paradigm in operation research, computer science and engineering.

Computation complexity theory has become pervasive, so much so that Garey and Johnson's comprehensive monograph discusses more than 300 combinatorial applications (320 application exactly), and the Travelling Salesman Problem is the first representative problem discussed. Consequently, the Travelling Salesman Problem would appear to be both a source of inspiration and a prime candidate for analysis by heuristic methods.

Cumulatively, this fertile decade of 1970's research has yielded much improved capabilities for applying optimization methods to combinatorial problems, capabilities that tend to counterbalance he trend, stimulated by computational complexity theory, towards heuristic methods. As a consequence, heuristic methods provide excellent opportunities for the current algorithmic developments for the Travelling Salesman Problem of the 1980's.

1.4. Methodology:

Currently, the search for faster heuristic methods for combinatorial optimization problems seems to follow two different directions. On the one hand, he search for faster computation machinery such as the FTTP, Pentium, and Quad core processor has received a substantial amount of attention. On the other hand, there has been a considerable amount of effort devoted to the development of better and efficient algorithms. The number of instances in which these methodologies and algorithms thus developed to solve the Travelling Salesman Problem have been used successfully in practical applications has been growing encouragingly over the past years. These algorithms are classified into two categories, namely exact and heuristic (approximate) algorithms.

An example of such exact algorithm for solving the Travelling Salesman Problem is the branch and bound algorithm. This method is generalized scheme for limiting the necessary enumeration

of the entire configuration space of the Travelling Salesman Problem, thus improving on the exhaustive search technique. It accomplishes this by arranging the configuration space as a tree and attempting to find the bounds by which entire branches of the configuration space may be discarded from the search. Let us consider a configuration space of an N city Travelling Salesman Problem, which may be represented by a number of binary state variables corresponding o the presence of a tour edge (or the branch of the tour); each edge directly connects city i to city j. From this configuration space, we can derive that there are N(N-1)/2 such possible edges, and of course "only" (N- 1)!/2 combinations of the binary state variables to map to the valid tours. It is important to note that, in this method, a tree is constructed such that it branches in two directions at each node, depending on whether or not a particular edge is considered part of the tour. As we descend through the tree, the distance of the current incomplete tour grows as certain edges considered are included in the tour. If a certain upper bound has already been established for the optimal tour length, then an entire of the configuration space tree may be eliminated if the current incomplete tour length already exceeds that bound. Equally important to know that as the algorithm proceeds through the search tree, lower and upper bounds may be discovered as new branches are traversed. Although this ability to prune the search tree is vital to the success of the branch and bound algorithm, such expansion and pruning of the search tree can continue endlessly.

Though most of these algorithms have aimed for efficiency and computational tractability, the TSP is a NP-complete problem. In other words, the TSP is unlikely to be solvable exactly by any algorithm or amount of computation time when the problem size (the number of cities in the TSP), N, is large. Because of the exponentially dependent nature of the computation on N, the computing time in employing an exhaustive search in solving the TSP for the exact solution is practically infeasible. Such infeasibility can be evidently demonstrated in a simple example. Let us consider a computer that can be programmed to enumerate all the possible tours for a set of N cities, keeping track of the shortest tour. Suppose this computer enumerates and examines a tour in one microsecond. At this rate the computer would solve a ten city problem in 0.18 seconds, which is not too bad. In a fifteen city problem, the computational effort would require over twelve hours. But, a twenty city problem would require nearly two thousand years. It is not too difficult to

render such an algorithm entirely impractical--only a small increase in the problem size causes a nearly unbounded computation time.

Because of the impracticality in employing exact algorithms in solving the Travelling Salesman Problem, there exists fortunately another algorithmic category that constitutes quite a few practical heuristic (or approximate) algorithms, known as iterative improvement algorithms, for finding nearly optimal solutions for the Travelling Salesman Problem and other combinatorial optimization problems.

Iterative improvement starts with a feasible tour and seeks to improve the tour via a sequence of interchanges. In other words, it begins by selecting an initial state in the space and successively applying a set of rules for alternating the configuration so as to increase the optimality of the current solution. Given a random starting solution, the algorithm descends on the surface of the objective function until it terminates at a local minimum. The local minimum occurs because none of the allowed transitions or moves in the configuration space yield states with lower objective function. Thus, one application of this algorithm yields what may be a fairly optimal solution. By repeating this procedure many times, the probability of finding more highly optimal states is increased. The best-known algorithms of this type are the edge (branch) exchange. In the general case, r edges in a feasible tour are exchanged for r edges not in that solution as long as the result remains a tour whose length, distance, or cost is less than that of the previous tour. Exchange algorithms are referred to as r-opt algorithms where r is the number of edges exchanged at each iteration.

In an r-opt algorithm, all exchanges of r edges are tested until there is no feasible exchange that improves the current solution. This solution is then said to be r-optimal. In general, the larger the value of r, the more likely it is that the final solution is optimal. Even for approximate algorithms, the number of operations necessary to test all r exchanges unfortunately increases rapidly as the number of cities increases. As a result, values of $r = 2$ and $r = 3$ are the ones most commonly used.

Such heuristic algorithms, whose rate of growth of the computation time is a low order polynomial in N, rather than exponential in N, have been observed to perform well. Among

these heuristic algorithms, a modified iterative improvement heuristic known as "Simulated Annealing" Algorithm is selected to investigate the "Traveling Salesman Problem".

1.5. Simulated Annealing:

Annealing is the process of heating a solid and cooling it slowly so as to remove strain and crystal imperfections. The Simulated Annealing process [2] [3] consists of first "melting" the system being optimized at a high effective temperature, then lowering the temperature by slow stages until the system "freezes" and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state. The sequence of temperatures attempted to reach to a steady-state equilibrium is referred to as an annealing schedule. During this annealing process, the free energy of the solid is minimized. The initial heating is necessary to avoid becoming trapped in a local minimum. Virtually every function can be viewed as the free energy of some system and thus studying and imitating how nature reaches a minimum during the annealing process should yield optimization algorithms.

In 1982, Kirkpatrick, Gelatt & Vecchi [2] observed that there is an analogy between combinatorial optimization problems such as the TSP and large physical systems of the kind studied in statistical mechanics. Using the cost function in place of the energy function and defining configurations by a set of energy states, it is possible, with the Metropolis procedure [4] [5] that allows uphill transitions, to generate a population of configurations of a given optimization problem at some effective temperature schedule. This temperature schedule [6] is simply a control parameter in the same units as the cost function. This procedure is highly inspired from "Markov Chain" [7].

Just what simulation, i.e. imitation, here means mathematically, along with its underlying relation with statistical physics. The resulting method called "Simulated Annealing" [8] [9], which is a heuristic combinatorial optimization technique that modifies the iterative improvement method by allowing the possibility of uphill moves in the configuration space, has become a remarkably powerful tool in solving global optimization problems in general and the TSP in particular.

1.6. Overview of the Thesis:

In chapter1, the problem statement for "Travelling Salesman Problem" has been given and some methodologies were also discussed. In chapter 2, theory related to "Simulated Annealing Algorithm" was given in relation with combinatorial optimization. In chapter 3, the Analytical Analysis of "Simulated Annealing Algorithm" along with its convergence mathematics was provided. In chapter 4, some practical applications available at laboratory were discussed in brief along with their "State Space Modeling". In chapter 5, simulation results obtained from application of "Simulated Annealing Algorithm" as well as "Tabu Search Algorithm" on "Travelling Salesman Problem" were given and in the last section a comparative analysis of "Simulated Annealing Algorithm" and "Tabu Search Algorithm" has been provided for 30, 50, 75, 442, 535 cities data collected from "stsp_v61 from math work provided by Arvind Seshadri" [10]. In the last chapter that is chapter 6, conclusion and future application of Simulated annealing algorithm were given.

CHAPTER 2

CLASSICAL SIMULATED ANNEALING ALGORITHM

2.1. Introduction

As briefly introduced in the previous chapter, Simulated Annealing [2] and independently [3] is one of the most powerful heuristic optimization techniques for solving difficult combinatorial optimization problems which have been known to belong to the class of NP-complete problems. This new approach was originally invented and developed by physicists based on ideas from statistical mechanics and motivated by an analogy to the behavior of physical systems in the presence of a heat bath. Because the number of molecules in the physical system of interest is very large, experimental measurements of the energy of every molecule in the system is practically impossible. Physicists were thus forced to develop statistical methods to describe the probable internal behavior of molecules.

In its original form, the Simulated Annealing Algorithm is based on the analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems, where the configurations actually are states (in an idealized model of a physical system), and the cost function is the amount of (magnetic) energy in a state. For this reason, the algorithm became known as "Simulated Annealing ". With the Metropolis procedure, Simulated Annealing offers a mechanism for accepting increases in the objective function in a controlled fashion. At each temperature setting, an increase in the tour length is accepted with a certain probability while a decrease in the tour length is always accepted. In this way, it is possible that accepting an increase will reveal a new configuration that will avoid a local minimum or at least a bad local minimum. The effect of the method is that one descends slowly. By controlling these probabilities, through the temperatures, many random starting configurations are in essence simulated in a controlled fashion. An analogy similar to this is well-known in statistical mechanics.

The non-physicist, however, can view it simply as an enhanced version of the familiar technique of "iterative improvement," in which an initial configuration is repeatedly improved by making small local alterations until no such alteration yields a better configuration. Simulated Annealing randomizes this procedure in such a way that allows for occasional "uphill moves," changes that worsen the configurations, in an attempt to reduce the probability of getting stuck at a poor and locally optimal configuration. Since the Simulated Annealing Algorithm is a generalization of "iterative improvement" and because of its apparent ability to avoid poor local optima, it can readily be adapted in solving new combinatorial optimization problems, thus, offering hope of obtaining significantly better results.

Ever since Kirkpatrick [2] introduced the concepts of annealing with incorporation of the Metropolis [4] procedure into the field of combinatorial optimization and applied it successfully to the "Ising spin class" problem, much attention has been devoted to the research of the theory and applications of Simulated Annealing. Important fields as diverse as VLSI design [2], and pattern recognition [5] have been applying Simulated Annealing with substantial success.

Computational results to date have been mixed. For further detailed examinations, an interested reader is encouraged to refer to [2].

In order to fully appreciate the thrust that is underlying the Simulated Annealing Algorithm as introduced in Section 2.4, it is important to understand Local Optimization which is briefly reviewed in Section 2.2 and the birth of the Simulated Annealing Algorithm which is discussed in Section 2.3.

2.2. Local Optimization

To gain a real appreciation of the Simulated Annealing Algorithm as will be described in more detail in Section 2.3 and Section 2.4, one must first understand Local Optimization. A combinatorial optimization problem can be specified by identifying a set of configurations together with a "cost function" that assigns a numerical value to each configuration. An optimal configuration is a configuration with the minimum possible cost (there may be more than

one such configuration). Given an arbitrary configuration to such a problem, Local Optimization attempts to improve on that configuration by a series of incremental, local changes. To define a Local Optimization algorithm, one first specifies a method for perturbing configurations so as to obtain different ones. The set of configurations that can be obtained in one such step from a given configuration i is called the neighborhood of i. The algorithm then performs the simple loop shown in Figure 2.1 (with the specific methods for choosing i and j left as implementation details).

Although i need not be a global optimal configuration when the loop is finally exited, it will be locally optimal in that none of its neighbors has lower cost. The hope is that "locally optimal" will be good enough. Because the locally optimal configuration is not always sufficient as can be seen from Figure 2.2, the Simulated Annealing Algorithm may provide the means to find both good locally optimal configurations and possibly a globally optimal configuration. Hence, it is the topic of discussion of the next section and the following.

1. Get an initial configuration i.
2. While (there is an untested neighbor of i) do the following:
   2.1 Let j be an untested neighbor of i.
   2.2 If cost (j) < cost (i), set i = j.
3. Return i.

Figure 2.1: Local Optimization Algorithm.

2.3 Statistical Mechanics- A Physical Analogy

As will be seen in the next section, Simulated Annealing is the algorithmic counterpart to a physical annealing process of statistical mechanics, using the well-known Metropolis Algorithm as its inner loop. Statistical mechanics concerns itself with analyzing aggregate properties of large numbers of atoms in liquids or solids. The behavior is characterized

by random numbers fluctuating about a most probable behavior, namely the average behavior of the system at that temperature. An important question is: What happens to the molecules in the system at extremely low temperatures, i.e. about zero degree? The low-temperature state may be referred to as the ground state or the lowest energy state of the system. Since low-temperature states are very rare, experiments that reveal the low-

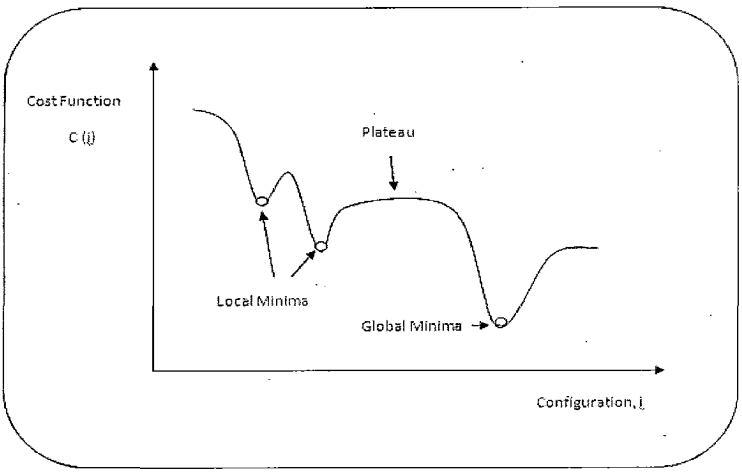

Figure 2.2: Plateau Local Minima and Global Minimum for the Cost Function.

temperature state of a material are performed by a process referred to as annealing. In condensed matter physics, annealing denotes a physical process in which a solid material under study in a heat bath is first melted by increasing the temperature of the heat bath to a maximum value at which all particles of the solid randomly arrange themselves in the liquid phase; this melted material is then cooled slowly by gradually lowering the temperature of the heat bath, with a long time spent at temperature near the freezing point.

It is important to note that the period of time at each temperature must be sufficiently long to allow a thermal equilibrium to be achieved; otherwise, certain random fluctuations will be frozen into the material and the true low-energy state or ground state energy will not be reached. The process is like growing a crystal from a melt. To simulate the evolution of the thermal equilibrium at any given temperature T, Metropolis [4] introduced a Monte Carlo method, a simple algorithm that can be used both to generate sequences of internal configurations or states and to provide an efficient simulation of collections of atoms in order to examine the behavior of gases in the presence of an external heat bath at a fixed temperature (here the energies of the individual gas molecules are presumed to jump randomly from level to level in line with the computed probabilities). In each step of this algorithm, a randomly generated atom is given a small displacement, and the resulting change, ΔE, in the energy of the system between the current configuration and the perturbed configuration is computed. If $\Delta E < 0$, the displacement is accepted, and the configuration with the displaced atom is used as the starting point of the next step. The case $\Delta E > 0$ is treated probabilistically: the probability that the configuration is accepted is $P(\Delta E) = \exp(-E / K_B T)$. This acceptance rule of the new configurations is known as the Metropolis criterion. Random numbers uniformly distributed in the interval (0,1) are a convenient means of implementing the random part of the algorithm. One such number is selected and compared with P(ΔE); if this random number is less than P(ΔE), then the new configuration is retained for the next step; otherwise, the original configuration is used to start the next step. By repeating the basic step many times and using the above acceptance criterion, one simulates the thermal motion of the atoms of a solid in thermal contact with a heat bath at each temperature T, thus allowing the solid to reach thermal equilibrium. This choice of P (ΔE) has the consequence that the system in a given state i with energy $E(i)$ evolve into the Boltzmann distribution.

$$P_T(i) = \exp(-E(i) / K_B T) / Z(T) \qquad (2.1)$$

Where

$Z(T)$ is a normalization factor, known as the partition function,

T is the temperature,

K_B is the Boltzmann constant,

i is a configuration of molecules in a system,

$E(i)$ is the energy of configuration i,

$\exp(-E / K_B T)$ is known as the Boltzmann factor,

and, $P_T(i)$ is its probability.

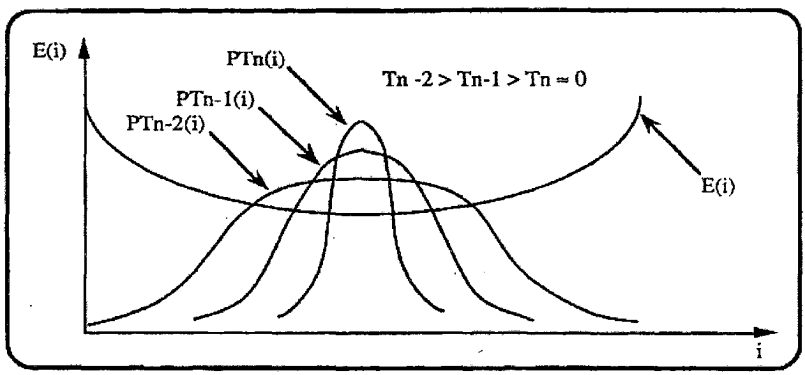

Figure 2.3: Boltzmann's distribution curve for an energy function at various Temperatures

Note that, as the temperature decreases, the Boltzmann distribution concentrates on states with the lowest energy, and finally when the temperature approaches zero, only the minimum energy states have a non-zero probability of occurrence.

In statistical mechanics, this Monte Carlo method, which is the Metropolis Algorithm, is a well-known method used to estimate averages or integrals by means of random sampling techniques. The general structure of the Metropolis Algorithm is summarized in Figure 2.4.

It is important to note that a decrease (downhill) in the change of energy is always accepted while an increase (uphill) in the change of energy is accepted probabilistically. After many

iterations of the Metropolis Algorithm, it is expected that the configuration of atoms would vary according to its stationary probability distribution.

The type of acceptance probability used for uphill moves in the Metropolis Algorithm may be used in the Simulated Annealing Algorithm. The ΔE of the Metropolis Algorithm is replaced by the change in the value of the objective function and the quantity $K_B T$ is replaced by the dimensionless version of the temperature, T. Given a sufficiently low temperature, the distribution of configurations of the optimization problem will converge to a Boltzmann distribution that sufficiently favors lower objective function states (the optimal states). The probability of accepting any uphill moves approaches zero as the temperature approaches zero. As a result, approaching thermal equilibrium requires an unacceptably large number of steps in the algorithm.

1. Generate an initial state i of the system.
2. Set the initial Temperature T > 0.
3. While (" Not yet frozen") do the following:
   3.1 While ("Not in thermal equilibrium") do the following:
   3.1.1 Perturb atom from state i to state j.
   3.1.2 Compute ΔE = Energy (j) – Energy (i)
   3.1.3 ΔE ≤ 0 * Decrease energy transition
   Then set i=j.
   3.1.4 ΔE > 0 * Increase energy transition
   Then set i=j with probability = $\exp(-E / K_B T)$
   3.2 Set T = Update (T) * Reduce Temperature
4. Return i *Return best state

Figure 2.4: General Metropolis Algorithm

The general approach of Simulated Annealing is to let the algorithm spend a sufficient Amount of time at a higher temperature, and is then slowly lowering the temperature by small incremental steps. The process is then repeated until a sufficiently low temperature has been obtained, i.e. T = **0.** This is faster than simply setting the temperature initially to a low

value and waiting for configurations of substances to reach thermal equilibrium. Annealing may be considered as the process of cooling slowly enough so that phase transitions are allowed to occur at their corresponding critical temperatures. Thus, to obtain pure crystalline systems, the cooling phase of the annealing process must proceed slowly while the system freezes.

However, it is well known [2] that if the cooling is too rapid, i.e. if the solid or crystal structure is not allowed to reach thermal equilibrium for each temperature value, defects and widespread irregularities or non-equilibrium states can be 'frozen' or locked into the solid, and meta stable amorphous structures corresponding to glasses can result rather than the low energy crystalline lattice structure. Furthermore, this process is known in condensed matter physics as "rapid quenching"; the temperature of the heat bath is lowered instantaneously, which results in a freezing of the particles in the solid into one of the meta stable amorphous structures. The resulting energy level would be much higher than it would be in a perfectly structured crystal. This "rapid quenching" process can be viewed as analogous to Local Optimization. When crystals are grown in practice, the danger of bad "local optima" is avoided because the temperature is lowered in a much more gradual way, by a process that Kirkpatrick calls "careful annealing". In this process, the temperature descends slowly through a series of levels, each held long enough for the crystal melt to reach "equilibrium" at that temperature. As long as the temperature is nonzero, uphill moves remain possible. By keeping the temperature from getting too far ahead of the current equilibrium energy level, we can hope to avoid local optima until we are relatively close to the ground state.

The correspondent analogy we are seeking now presents itself. Each feasible configuration of the combinatorial optimization problem or each feasible tour of the TSP corresponds to a state of the system; the configuration space of the combinatorial optimization problem or the permutation space of the TSP corresponds to the state space of the system; the cost or objective function corresponds to the energy function; the objective value associated with each feasible tour corresponds to the energy value associated with each state of that system; the optimal configuration or tour associated with the optimal cost value corresponds to the ground state associated with the lowest energy value of the state of the physical system. The

analogy is summarized in Figure 2.5.

Physical System	Optimization Problem	Travelling Salesman
State	Feasible Configuration	Feasible Tour
State Space	Configuration Space	Permutation Space
Ground State	Optimal Configuration	Optimal Tour
Energy Function	Cost Function	Cost Function
Energy	Cost	Cost
Rapid Quenching	Local Optimization	Local Optimization
Careful Annealing	Simulated Annealing	Simulated Annealing

Figure 2.5: Analogy between Physical System and Combinatorial Optimization

2.4 Classical Simulated Annealing

As was discussed in Section 2.2 and illustrated by Figure 2.2, the difficulty with Local Optimization is that it has no way to "back out" of the unattractive local optima because it never moves to a new configuration unless the direction is "downhill," i.e. to a better value of the cost function, Simulated Annealing is an approach that attempts to avoid the entrapment in poor local optima by allowing an occasional "uphill" move. This is done under the influence of a random number generator and an annealing schedule. The attractiveness of using the Simulated Annealing approach for combinatorial optimization problems is that transitions away from a local optimum are always possible when the temperature is nonzero. As pointed out by Kirkpatrick [2], the temperature is merely a control parameter; this parameter controls the probability of accepting a tour length such, it is expressed in the same units as the objective function. In implementing the approach, any improvement procedure could be used.

As was seen, the Metropolis Algorithm can also be used to generate sequences of

configurations of a combinatorial optimization problem. In that case, the configurations assume the role of the states of a solid while the cost function C and the control parameter called the annealing schedule, T, take the roles of energy and the product of temperature and Boltzmann's Constant, respectively. The Simulated Annealing Algorithm can now be viewed as a sequence of Metropolis Algorithms evaluated at each value of the decreasing Sequence of annealing schedule, which is defined to be $T = \{t_1, t_2, ..., t_n\}$, where $t_1 > t_2 > .... > t_{n-1} > t_n$. It can thus be described as follows. Initially the annealing schedule has given a high value, and a sequence of configurations of the combinatorial optimization problem is generated. As in the iterative improvement algorithm, a generation mechanism is defined, so that, given a configuration i, another configuration j can be obtained by choosing at random a configuration from the neighborhood of i. The latter corresponds to the small perturbation in the Metropolis Algorithm. Let $\Delta C(i, j) = C(j) - C(i)$, then the probability for configuration j to be the next configuration in the sequence is given by 1 if $\Delta C(i, j) \leq 0$, and by $\exp(-\Delta C(i, j) / T)$, if $\Delta C(i, j) > 0$ (Metropolis Criteria). Thus, there is a non-zero probability of continuing with a configuration with higher cost than the current configuration. This process is continued until equilibrium is reached, i.e. until the probability distribution of the configuration approaches the Boltzmann distribution, now given by

$$P_T\{configuration = i\} = q_i(T) = \exp(-C(i) / T) / Q(T),$$

Where Q(T) is the normalization constant depending on the annealing schedule T, which is equivalent to partition function Z(T).

The probability distribution curve for the cost function is analogous to Figure 2.3 with E(i) is replaced by C(i).

The annealing schedule T is then lowered in incremental steps, with the system being allowed to approach equilibrium for each step. The algorithm is terminated for some small value of T, at which virtually no further deteriorations or increases in cost are accepted. The final 'frozen' configuration is then taken as the optimal configuration of the problem under consideration. The main steps in the Simulated Annealing Algorithm are outlined in Figure 2.6.

1. Generate an initial random configuration i.
2. Set the initial temperature T > 0.
3. While (not yet "frozen") do the following:
   3.1. While ("inner loop iteration" not yet satisfied) do the following:
      3.1.1. Select the random neighbor j from configuration i.
      3.1.2. Compute $\Delta C(i, j)$ = Cost(j) – Cost(i);
      3.1.3. If $\Delta C(i, j) \leq 0$ * Downhill transition
         Then i = j.
      3.1.4 If $\Delta C(i, j) > 0$ * Uphill Transition
         Then set i = j with probability = $\exp(-\Delta C(i, j) / T)$
   3.2. Set T = Update (T) * Reduce the Temperature
4. Return i. * Return Best Configuration

Figure 2.6: Simulated Annealing Algorithm

Thus, as with iterative improvement, we have again a generally applicable approximation algorithm: once configurations, a cost function and a generation mechanism or, equivalently, a neighborhood structure) are defined, a combinatorial optimization problem can be solved along the lines given by the description of the Simulated Annealing Algorithm. The heart of this procedure is the loop at Step 3.1. Note that the acceptance criterion is implemented by drawing random numbers from a uniform distribution on (0,1) and comparing these with $\exp(-\Delta C(i, j) / T)$. Note also that $\exp(-\Delta C(i, j) / T)$ will be a number in the interval (0,1)

when ΔC and T are positive, and so can rightfully be interpreted as a probability. Note also how this probability depends on ΔC and T. The probability that an uphill move of size ΔC will be accepted diminishes as the temperature declines, and, for a fixed temperature T, small uphill moves have higher probabilities of acceptance than larger ones. This particular method of operation is motivated by a physical analogy of the physics of crystal growth described in the last section.

The main difference between the Simulated Annealing Algorithm and the Metropolis Algorithm is that the Simulated Annealing Algorithm iterates with variable temperature while the Metropolis Algorithm iterates with a constant temperature. As the temperature is slowly decreased to zero or annealed, the system approaches to steady state equilibrium. This implies that the cost function should converge to a global minimum. It is worthy to emphasize that the cooling or annealing process should be done slowly; otherwise, the system can get stuck at a local minimum.

Ever since Kirkpatrick had recognized the physical analogy between statistical mechanics and combinatorial optimization, the Simulated Annealing Algorithm has been important in many disciplines. Not only has it been successfully applied in many important fields of science and engineering but also it has been one of the major stimulants of research in the academic and industrial communities. The force that makes the Simulated Annealing Algorithm powerful is its inherent ability to avoid and/or to escape from being entrapped at local minima, which are so many for a medium-size combinatorial optimization problem in general and the TSP in particular.

In this chapter, the underlying motivation and historical development of the Simulated Annealing Algorithm has been covered. To provide some useful results for the subsequent chapters, a mathematical model and a quantitative analysis of the Simulated Annealing Algorithm are studied in the next chapter.

CHAPTER 3

QUANTITAIVE ANALYSIS OF SIMULATED ANNEALING ALGORITHM

3.1 Introduction

In Chapter **1,** a brief description of the Simulated Annealing was introduced. In Chapter 2, the origin and the motivation of Simulated Annealing were examined in detail and the Algorithm was outlined. In this chapter, certain key mathematical concepts which are the underlying foundation of Simulated Annealing will be investigated.

The Simulated Annealing Algorithm can be modeled mathematically by using concepts of the theory of Markov chains [7]. Since a detailed analysis of these Markov chains is beyond the scope of this thesis, they are extensively discussed and proved by a number of authors [7] that under certain conditions, the algorithm converges asymptotically to an optimal solution. Thus, asymptotically, the algorithm is an optimization algorithm. In practical applications, however, asymptoticity is never attained and thus convergence to an optimal solution is no longer guaranteed. Consequently, in practice, the algorithm is an approximate algorithm.

The performance analysis of an approximate algorithm concentrates on following two quantities:

*The quality of the final solution obtained by the algorithm, i.e. the difference in cost value between the final solution and a globally minimal configuration;

*and, the running time required by the algorithm.

For the Simulated Annealing Algorithm, these quantities depend on the problem instance as well as the annealing schedules.

Traditionally, three different types of performance analysis are distinguished, namely *worst*-case analysis, average-case analysis, and empirical analysis. The worst-case analysis is concerned with upper bounds on quality of the final solutions, i.e. how far from optimal the

constructed tour can be, while the average-case analysis is focused on the expected values of quality of the final solutions and running times for a given probability distribution of the problem instances. Empirical analysis here means the analysis originating in or based on computational experience. In other words, solving many different instances of the TSP with different annealing schedules and drawing conclusions from the results, with respect to both quality of solutions and running time. In this way, the effects of the annealing schedules on the algorithm can be analyzed. It is interesting to analyze these effects because, even for a fixed instance, the computation time and the quality of the final solution are random variables, due to the probabilistic nature of the algorithm. All three approaches are attempts to provide the information that will help in answering the question 'How well will the algorithm perform (how near to optimal will be the tours it constructs) on the problem instances.' Each approach has its advantages and its drawbacks.

Worst-case analysis can provide guarantees that hold for individual instances and does not involve the assumption of any probability distribution. The drawback here is that, since the guarantee must hold for all instances, even ones that may be quite atypical, there may be a considerable discrepancy in the behavior of an algorithm. Empirical analysis can be most appropriate if the problem instances on which it is based are similar to the problem of interest. It may be quite misleading if care is not taken in the choice of test problems, or if the test problems chosen have very different characteristics from those at hand. Average- case (or average ensemble) analysis can tell us a lot, especially when we will be applying the algorithm to many instances having similar characteristics. However, by its nature, this type of analysis must make assumptions about the probability distribution on the class of instances, and if the assumptions are not appropriate then the results of the analysis may not be germane to the instances at hand.

A final problem with worst case and average case analysis of heuristics comes from the rigorous nature of both approaches. Analyzing a heuristic in either way can be challenging mathematical task. Heuristics that yields nice probabilistic bounds may be inappropriate for worst case analysis, and the heuristics that behaves well in worst case analysis are often exceedingly difficult to analyze probabilistically. In addition, many heuristics do not seem to be susceptible to either type of analysis.

When studying the simulated annealing algorithm, an additional probabilistic aspect is added to

the above classification. Besides the probability distribution over the set of problem instances, there is also a probability distribution over the set of possible solution for a given problem. Thus, in the average case analysis, the average can be referred to as the average of a set of solutions of a given problem instance.

In this chapter, a combination of both the average case analysis (average ensemble) for the set of solution for the given problem instance and empirical analysis is grouped as "semi-empirical" average case analysis will be investigated for two representative instances of the Travelling Salesman Problem. Using these instances to present a "semi-empirical" average case analysis of the algorithm by running it number of times, it is possible to reproduce the observed behavior by using standard technique from statistical physics and some assumptions on configuration density. Presently a systematic investigation of the typical behavior and the average case performance analysis of Simulated Annealing Algorithm remain as an open research problem.

In section 3.2, the core mathematical model of Simulated Annealing Algorithm based on Markov chains is represented and discussed. In this section, the salient features of annealing schedule which will be useful in computation study are also highlighted. And, the analysis of the cost function is presented in section 3.3.

3.2 Mathematical Model:

A combinatorial optimization problem can be characterized by configuration space $\Re$, denoting the set of all possible configuration i, and the cost function $C: \Re \rightarrow R$, which assigns a real number C(i) to each configuration i. C is assumed to be defined such that the lower value of C, better the corresponding configuration, with respect to Optimization criteria. This can be done without the loss of the generality. The objective is to find the optimal configuration i* for which

$$C(i^*) = C_{\min} = \min\{C(i) \mid i \in \Re\} \qquad 3.1$$

Where $C_{\min}$ denotes the minimum cost.

To apply the Simulated Annealing Algorithm, a mechanism known as neighborhood structure or

the perturbation function is used o generate a new configuration, i.e. a neighborhood of i, by small perturbation. A neighborhood j defined as the set of configurations that can be reached from configuration i by a single perturbation. The Simulated Annealing algorithm starts off with a given initial configuration and continuously tries to transform a current configuration into one of its neighbor by applying a perturbation mechanism and an acceptance criterion. The acceptance criterion allow for deteriorations in the cost function, thus enabling the algorithm to escape from local minima.

3.2.1 Asymptotic Convergence:

As mentioned in the last section, the Simulated Annealing Algorithm can be formulated as a sequence of Markov chains, each Markov chain being a sequence of trials whose outcomes X1,X2, X3,…., satisfy the following two properties:

(1) Each outcomes belongs to a finite set of outcomes {1,2,3,…,N} called the configuration space $\Re$ of the system; if the outcome of the k^{th} trial is i, then the system is said to be in state i, at time k or at the k^{th} step.

(2) The outcome of any trial depends at most of the immediately preceding trial and not upon any other previous outcome; i.e. the outcome is only dependent on the outcome of previous trial, with each pair of states or configurations (i,j) there is given the probability P_{ij} such that j occurs immediately after i occurs.

Such a stochastic process is known as (Finite) Markov chain. The numbers P_{ij} is called the transition probabilities that can be arranged into a Transition matrix P below.

$$P = \begin{pmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{pmatrix}$$

called the transition matrix.

Thus, with each configuration i, there corresponds the i^{th} row $(P_{i1}, P_{i2}, ..., P_{in})$ of the transition

matrix P; if the system is in configuration i, then this row vector represents the probabilities of all possible outcomes of the next trial and so it is the probability vector, whose row sum is always equals to one.

Note that the outcomes of the trials here are the configuration. For example, the outcome of the given trial is perturbed configuration j while the outcome of the previous trial is the current configuration i. So, Markov chain is described by means of a set of conditional probabilities $P_{ij}(k-1,k)$ for each pair of outcomes (i,j); $P_{ij}(k-1,k)$ is the probability that the outcome of the k^{th} trial is j, given that the outcome of the $(k-1)^{th}$ trial is i. Let $a_i(k)$ denote the probability of outcome i at the k^{th} trial, then $a_i(k)$ is obtained by solving the recursive relation:

$$a_i(k) = \sum_i a_i(k-1).P_{ii}(k-1,k)\ ,\ \mathrm{k} = 1,2,\ldots, \tag{3.2}$$

where the sum is taken over all possible outcomes.

Let X(k) denotes the outcome of the k^{th} trial. Then,

$$P_{ij}(k-1,k) = P_T\{X(k) = j \mid X(k-1) = i\} \tag{3.3}$$

And

$$a_i(k) = P_T\{X(k) = i\} \tag{3.4}$$

If the conditional probabilities depend on k, the corresponding Markov chain is called homogeneous, otherwise it is called inhomogeneous.

In the case of Simulated Annealing Algorithm, the conditional probability $P_{ij}(k-1,k)$ denotes the probability that the k^{th} transition is from configuration i to configuration j. Thus, X (k) is the configuration obtained after k transitions. In this view, $P_{ij}(k-1,k)$ is the transition probability and the $|\Re| \times |\Re|$ matrix $P(k-1,k)$ the transition matrix.

The transition probabilities depend on the value of the annealing schedule T. Thus, if T is kept constant, the corresponding Markov chain is homogeneous, and its transition probabilities, i.e. the probability that a trial transforms configuration i into configuration j, is defined as:

$$P_{ij}(T) = \begin{cases} A_{ij}(T)G_{ij}(T) & if \quad i \neq j \\ 1 - \sum_{k \in \Re, k \neq i} A_{ik}(T)G_{ik}(T) & if \quad i = j \end{cases} \tag{3.5}$$

Where

- $P_{ij}(T)$ denotes the transition probabilities.
- $G_{ij}(T)$ denotes the gencration probability, i.e. the probability of generating configuration j from configuration i.
- $A_{ij}(T)$ denotes the acceptance probability, i.e. the probability of accepting configuration j given the configuration i and j.
- And T is the Annealing schedule.

Each transition probability is defined as a product of following two probabilities: the generation probability $G_{ij}(T)$ of generating configuration j from configuration i, and the acceptance probability $A_{ij}(T)$ of accepting configuration j, once it has been generated from configuration i. The corresponding matrix G(T) and A(T) are called the generation and acceptance matrices, respectively. As a result of definition in equation 3.5, P(T) is stochastic matrix, i.e. $\forall i : \sum_j P_{ij}(T) = 1$.

- A homogeneous algorithm: The algorithm is described by the sequence of homogeneous Markov chains. Each Markov chain is generated at a fixed value of T and T is decreased in between the subsequent Markov chains, and
- An inhomogeneous algorithm: The algorithm is described by a single inhomogeneous Markov chain. The value of T is decreased in between the subsequent transitions.

The Simulated Annealing Algorithm obtains a global minimum if after a large number of transitions, K, i.e. $K \to \infty$, the following relation holds:

$$P_T\{X(K) \in \Re_{opt}\} = 1, \tag{3.6}$$

Where $\Re_{opt}$ is the set of globally minimal configurations.

Equation (3.6) can be proved under a number of conditions on probabilities $G_{ij}(T)$ and $A_{ij}(T)$; asymptotically; i.e. for infinitely long Markov chains and $T \to 0$, the algorithm finds an optimal configuration with probability equal to one. Let X(K) denotes outcome of the k^{th} trial of a Markov chain; i.e. under the condition that the Markov chain is irreducible, periodic and recurrent, there exist a unique equilibrium distribution given by $|\Re|$ vector q(T). The component qi(T) denotes the probability that the configuration i will be found after infinite number of trials and are given by the following expression:

$$q_i(T) = \lim_{k\to\infty} P_T\{X(k) = i / T\} = \lim_{k\to\infty}([P^k(T)]^T a)_i \tag{3.7}$$

Where a denotes the initial probability distribution of the configuration and P(T) is the transition matrix, whose entries are given by $P_{ij}(T)$. Under certain additional conditions on the probabilities $G_{ij}(T)$ and $A_{ij}(T)$, the algorithm converges to $T \to 0$ to a uniform distribution on the set of optimal configuration, i.e,

$$\lim_{T\to 0}(\lim_{k\to\infty} P_T\{X(k) = i / T\}) = \lim_{T\to 0} q_i(T) = \pi_i \tag{3.8}$$

$$\pi_i = \begin{cases} |\Re_{opt}|^{-1} & if \quad i \in \Re_{opt} \\ 0 & elsewhere \end{cases} \tag{3.9}$$

Where $\Re_{opt}$ denotes the se of optimal configurations.

Here, we apply he standard form of the Simulated Annealing algorithm, i.e. the perturbation probability $G_{ij}(T)$ is chosen independent of T and uniformly over the neighborhood of a given configuration i. the acceptance probability is chosen as

$$A_{ij}(T) = \begin{cases} \exp(-\Delta C_{ij} / T) & if\ \Delta C_{ij} > 0 \\ 1 & if\ \Delta C_{ij} \le 0 \end{cases} \tag{3.10}$$

Where ΔC_{ij} = C(j) – C(i). For his choice of components of the equilibrium distribution take the form

$$q_i(T) = \frac{\exp\{[C_{\min} - C(i)]/T\}}{\sum_{j \in \Re} \exp\{[C_{\min} - C(i)]/T\}} \tag{3.11}$$

The above result is extremely useful when the cost function is analyzed.

3.2.2. Annealing Schedule:

As mentioned previously, the performance of the Simulated Annealing Algorithm is a function of the annealing schedules. Hence, it is common that one resorts to an implementation of the Simulated Annealing Algorithm in which a sequence of Markov chains of finite length is generated at decreasing values of the annealing schedule. Optimization is begun at a starting value of the temperature T_0 and continues by repeatedly generating Markov chains for decreasing values of T, until T approaches 0. This procedure is governed by the annealing schedule. Generally, the parameters used in studying the performance of the Simulated Annealing Algorithm are

(1) The length L of the individual Markov chains

(2) The stopping criteria for terminating the algorithm

(3) The start value T_0 of the Annealing schedule

(4) The decrement functions of the annealing schedule.

The salient features of these parameters are summarized here.

(1) Markov chain length L: All Markov chains are chosen equally long. In practice, the number of cities in the TSP tour or the number of runs of the algorithm is taken to be equal to the length of Markov chains.

(2) Stopping Criteria: There are many criteria for terminating the Simulated Annealing Algorithm presently existed. Here, the algorithm terminates at a certain maximum number of iterations arbitrarily set by the user.

(3) Starting Value T_0 : The purpose of the starting temperature value is to begin the thermal system at a high temperature. There are many variations of the annealing schedules. This starting value is as high as 2000 and as low as 20.

(4) Annealing Schedule T: The performance of Simulated Annealing Algorithm is a function of annealing schedule. Because of this dependence, the following two well known annealing schedules which proves to provide good solutions to the TSP by varying the parameter c and d, i.e. $0.9 < c < 0.99$ and $5 < d < 30$.

$$T_{k+1} = cT_k; \qquad k = 0,1,2,\ldots,\max\_iteration \tag{3.12}$$

And

$$T_k = d / \log k; k = 2,3,4,\ldots,\max\_iteration \tag{3.13}$$

Note that as a consequence of the asymptotic convergence of the Simulated Annealing Algorithm, it is intuitively clear that the slower the "cooling" is carried out, the larger the probability that the final configuration is close to an optimal configuration. Thus, the deviation of the final configuration from an optimal configuration can be made as small as desired by investing more computational effort. The literature has not elaborated on the probabilistic dependence on the parameters of the annealing schedule. In this chapter semi-empirical results on this topic are represented. A more theoretical treatment is still considered as an open research topic.

3.3. Analysis of Cost Function:

In this section, some quantitative aspects of the Simulated Annealing Algorithm are discussed. The discussion is based on an extensive set of numerical data obtained by applying the algorithm to a specific instance of the Traveling Salesman Problem. The behavior of the Simulated Annealing Algorithm is analyzed. In this section, an analytical approach to derive the expectation and the variance of the cost function in terms of the annealing schedule is analyzed. The discussion is based on an average-case performance analysis.

To model the behavior of the Simulated Annealing Algorithm, an analytical approach to calculate the expectation $\langle C \rangle_T$ and the variance $\sigma^2{}_T$ of the cost function is discussed. Let X denotes the outcome of a given trial; the $\langle C \rangle_T$ and $\sigma^2{}_T$ are defined as

$$\langle C \rangle_T = \sum_{i \in \Re} P_T\{X = i / T\} C(i) \tag{3.14}$$

And

$$\sigma^2{}_T = \sum_{i \in \Re} P_T\{X = i / T\} [C(i) - \langle C \rangle_T]^2 \tag{3.15}$$

In equilibrium we obtain, using equation 3.7 and 3.11,

$$\langle C \rangle_T = \sum_{i \in \Re} q_i(T) C(i) = \frac{\sum_{i \in \Re} \exp\{[C_{\min} - C(i)] / T\} C(i)}{\sum_{j \in \Re} \exp\{[C_{\min} - C(j)] / T\}} \tag{3.16}$$

And

$$\sigma^2{}_T = \sum_{i \in \Re} q_i(T) [C(i) - \langle C \rangle_T]^2 = \frac{\sum_{i \in \Re} \exp\{[C_{\min} - C(i)] / T\} [C(i) - \langle C \rangle_T]^2}{\sum_{j \in \Re} \exp\{[C_{\min} - C(j)] / T\}} \tag{3.17}$$

Next the configuration density $\omega(C)$ is defined as

$$\omega(C)dC = \frac{1}{|\Re|}|\{i \in \Re \mid C \le C(i) < C + dC\}| \tag{3.18}$$

Then in case of Simulated Annealing Algorithm employing the acceptance probability of equation 3.10, the equilibrium configuration density $\Omega(C,T)$ at a given value of T is given by

$$\Omega(C,T)dC = \frac{\omega(C)\exp[(C_{\min} - C)/T]dC}{\int_{-\infty}^{+\infty} \omega(C')\exp[(C_{\min} - C')/T]dC'} \tag{3.19}$$

Clearly, $\Omega(C,T)$ is the equivalent of the stationary distribution q(T) given by equation 3.11. As indicated by the notation "equilibrium", $\Omega(C,T)$ is the configuration density in equilibrium when applying the Simulated Annealing Algorithm. Thus, one obtains

$$\langle C \rangle_T = \int_{-\infty}^{\infty} C'\Omega(C',T)dC' \tag{3.20}$$

And

$$\sigma^2{}_T = \int_{-\infty}^{\infty} [C' - \langle C \rangle_T]^2 \Omega(C',T)dC' \tag{3.21}$$

Given an analytical expression for the configuration density $\omega(C)$, it is possible to evaluate the integral of the equations 3.19, 3.20, 3.21. To estimate $\omega(C)$ for a given combinatorial optimization problem is in most cases very hard. Indeed, $\omega(C)$ may vary drastically for different specific problem instances, especially for C values close to $C_{\min}$.

The average Cost $\overline{C}$ and the standard deviation $\sigma(T)$ of the cost as a function of the annealing schedule T when applying the Simulated Annealing Algorithm to an instance of the TSP are the following expressions,

$$\overline{C}(T) = L^{-1}\sum_{i=1}^{L} C_i(T) \tag{3.22}$$

And

$$\sigma(T) = \left\{ L^{-1}\sum_{i=1}^{L} [C_i(T) - \overline{C}(T)]^2 \right\}^{1/2} \tag{3.23}$$

Where the average is taken over the values of cost function $C_i(T)$, for i = 1,....,L, of the Markov chains generated at a given value of annealing schedule T. From the above relations, the behavior of Simulated Annealing Algorithm is observed for many problem instances [2]. Furthermore, some characteristic feature of the expectation $\langle C \rangle_T$ and the variance ${\sigma^2}_T$ of the cost function can be deduced. For large value of **T**, the average and standard deviation of the cost are about constant and are equal to $\overline{C}(\infty)$ and $\sigma(\infty)$. This behavior is directly obtained from Equations 3.16 and 3.17, or equations 3.18-3.21, namely

$$\langle C \rangle_\infty = \lim_{T\to\infty} \langle C \rangle_T = \frac{1}{|\Re|}\sum_{i\in\Re} C(i) \tag{3.24}$$

And

$${\sigma^2}_\infty = \lim_{T\to\infty} {\sigma^2}_T = \frac{1}{|\Re|}\sum_{i\in\Re} [C(i) - \langle C \rangle_\infty]^2 \tag{3.25}$$

Note that the more detailed estimate of the average case performance of the Simulated Annealing Algorithm can only be deduced from rigorous performance analysis which takes into account the detailed structure of the optimization problem at hand. Presently, such a theoretical average case performance analysis remains to be as an open research problem.

The average case performance of the Simulated Annealing Algorithm is discussed by analyzing the expectation and the variance of the cost function as a function of annealing schedule for a certain instance of the Traveling Salesman Problem; the results can be summarized as follows:

- The performance of the Simulated Annealing Algorithm depends strongly on the chosen

annealing schedule; this is especially true for the quality of solution obtained by the algorithm.

- With a properly chosen annealing schedule, near optimal solution may be obtained.

In this chapter, certain key mathematical concept which is underlying foundation of Simulated Annealing Algorithm was examined.

CHAPTER 4

APPLICATION OF SIMULATED ANNEALING ALGORITHM

In order to assess the performance of the optimization algorithms, various control engineering problems were considered. Some of the practical applications used are:

- Inverted Pendulum
- Ball and Beam System
- Magnetic Levitation System

4.1. Inverted Pendulum:

The inverted pendulum control problem [11] is usually presented as a pole balancing task. The system to be controlled consists of a cart and a rigid pole hinged to the top of the cart. The cart can move left or right on a one-dimensional bounded track, whereas the pole can swing in the vertical plane determined by the track. The linear system equations around $\theta = \pi$ in the state space are given by equation 5.1 and 5.2.

The state of the system is defined by values of four system variables:

- x , denotes the cart position
- $\dot{x}$, denotes the cart velocity
- θ, denotes the pendulum angle of the pendulum pole
- $\dot{\theta}$,denotes the angular velocity of the pendulum pole

Here, the controlling action is applied to the system to prevent the pendulum pole from falling from the specified position and at the same time to keep the cart within the specified limits of position.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-(I+ml^2)b}{I(M+m)+Ml^2} & \dfrac{m^2gl^2}{I(M+m)+Ml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-mlb}{I(M+m)+Ml^2} & \dfrac{mgl(M+m)}{I(M+m)+Ml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{(I+ml^2)}{I(M+m)+Ml^2} \\ 0 \\ \dfrac{ml}{I(M+m)+Ml^2} \end{bmatrix} u \tag{5.1}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \tag{5.2}$$

Where;

M = Mass of the cart = 0.5 kg

m = Mass of the pendulum = 0.2 kg

b = Friction of the cart = 0.1 N/m/sec

I = Inertia of the pendulum = 0.006 kgm^2

l = Length of the pendulum's center of mass

F = Force applied to the cart

4.2. Ball and Beam System:

The ball-beam system [12] is a frequently encountered example of nonlinear dynamical system. While the ideal system is indeed nonlinear, its practical implementation has additional non-linearity's, including: dead zone, backlash introduced by the DC motor and gearbox, discrete position sensing and uneven rolling surface.

The motion of the motor's shaft is governed by IPM100 intelligent drive. This is a high precision, fully digital servo drive with embedded intelligence and 100W power amplifier suitable for brushless/brush motors. Based on feedback information from sensors, it computes and then applies appropriate PWM modulated voltage to the motor windings in such a way that a sufficient torque moves the motor shaft according the programmed control algorithm. This embedded intelligence provides a true real-time control performance independent of any delays caused by Personal Computer's non-real time Operating System.

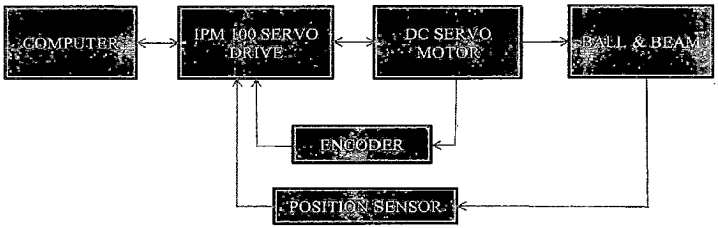

Figure: 4.1. Structure of control Strategy for Ball & Beam system

The closed loop control strategy employed for the application is given in Figure- 4.1. The DC motor provides actuation of the beam via a gear. The PID control algorithm inside IPM100 intelligent drive is employed in an inner control loop as a motor position controller. The PID gains are tuned in such a way that the motor exhibits a fast response without overshoot.

4.2.1. Mechanical Model of Ball and Beam System:

The rough figure of mechanical model of Ball & Beam system is shown in figure 4.2.

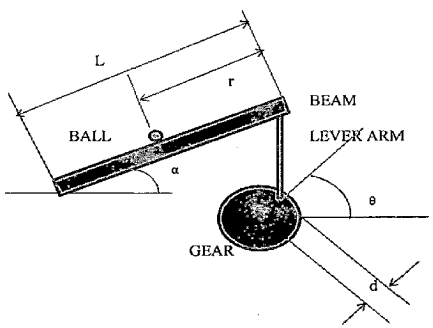

Figure: 4.2. Mechanical system of Ball and Beam

For the given system gear ratio is 107:25.

Let the angle between the line that connects the joint of the lever arm with the center of the gear, and the horizontal line be θ (there should be some boundaries on its range so that it can reach the safe maximum and minimum limits); the distance between the center of the gear and the joint of the lever arm be *d*, and the length of the beam be *L*. Then the beam angle α can be expressed in terms of the rotation angle of the gear θ according to the following equation:

$$\alpha = \frac{d}{L}\theta \tag{4.3}$$

In turn, as it has just been noted above, the angle θ is connected with the rotational angle of motor shaft through reduction gear ratio n=4.28. The controller design task is to keep the position of the ball r equal to the specified target position by properly manipulating the gear angle θ.

The dynamics of the ball is subjected to the gravity, inertial and centrifugal forces. The ball linear acceleration along the beam is given by the following simple equation [8]:

$$\left(\frac{J}{R^2} + m\right) r + mg\sin\alpha - m\dot{r}(\dot{\alpha})^2 = 0 \tag{4.4}$$

Where

g denotes the Gravitational acceleration

m denotes the mass of the ball

J denotes the moment of inertia of the ball

r denotes the position of the ball along the beam

R denotes the radius of the ball

In this, some assumptions are taken into consideration that the ball moves without slipping and friction between the ball and beam is negligible.

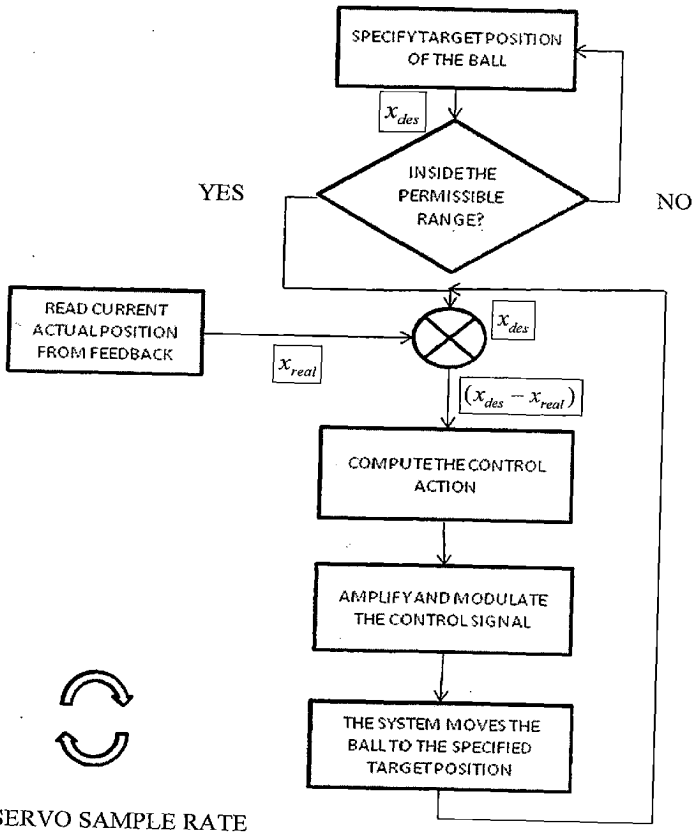

Figure: 4.3: Flowchart of Inherited control algorithm for Ball and Beam System

4.3. Magnetic Levitation System:

Magnetic Levitation System [13] works on principle of Electromagnetic Induction to control position of ball at required position. When current go through the winding, electromagnetic force F will be generated. By controlling the current in the electromagnet winding to balance the steel ball gravity force *mg* by magnetic force, the steel ball will be levitated in the air. Closed loop control is required for the stability and anti-interference. The distance x from the steel ball to electric magnet is detected by sensor system composed of light source and light sensor. To enhance the performance, the speed of the distance variance can also be considered. The control current is the input for magnetic levitation control object.

In system modeling, the input is control current of the electromagnet, the influence of inductance is not considered. Assume the power amplifier output current is strictly linear with input voltage without delay.

The system can be described by following equation:

$$m\frac{d^2x}{dt^2} = k_i(i-i_0) + k_x(x-x_0) = \frac{2Ki_0}{{x_0}^2}i - \frac{2K{i_0}^2}{{x_0}^3}x \tag{4.5}$$

After taking Laplace Transform:

$$x(s)s^2 = \frac{2Ki_0}{m{x_0}^2}i(s) - \frac{2K{i_0}^2}{m{x_0}^3}x(s) \tag{4.6}$$

From boundary equation $mg = -K(\frac{{i_0}^2}{{x_0}^2})$, the system open loop transfer function is:

$$\frac{x(s)}{i(s)} = \frac{-1}{As^2 - B} \tag{4.7}$$

Define the input variable as the input voltage of the power amplifier U_{in}, output variable as a output voltage U_{out} reflecting x (the voltage output of the process circuit at the back of the sensor), the system control object model can be expressed as:

$$G(s) = \frac{U_{out}}{U_{in}} = \frac{K_s x(s)}{K_a i(s)} = \frac{-(K_s / K_a)}{As^2 - B} \tag{4.8}$$

$$A = \frac{i_0}{2g} \tag{4.9}$$

$$B = \frac{i_0}{x_0} \tag{4.10}$$

The open loop system characteristic equation is:

$$As^2 - B = 0 \tag{4.11}$$

The system open loop pole is: $s = \pm\sqrt{\frac{B}{A}} = \pm\sqrt{\frac{2g}{x_0}}$

Then, the system state variables are: $x_1 = U_{out}, x_2 = \dot{U}_{out}$ and the system state equations as follows:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{2g}{x_0} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{2g.K_s}{i_0.K_a} \end{pmatrix} U_{in} \tag{4.12}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1 \tag{4.13}$$

There is an open loop pole at the right plane, by stability criterion; stable system should have all the open loop poles on the left plane. Therefore the GML system is essentially unstable.

In fact, the inductance of the coil will prevent the current from changing too fast; this effect cannot be ignored. Thus the current model is slightly different from the real case. To analyze the system accurately, voltage control model is also important. For real system, parameters are given as follows:

m = 22g, x_0^* =20.0 mm, Iron core Diameter = 22 mm, Enameled wire diameter = 0.8 mm,

R = 13.8 Ω, r = 12.5 mm (radius of the ball), N= 2450 circles, K= 2.3142e-004Nm^2/A^2, i_0^*=0.6105 A, K_f=0.25.

Therefore, the transfer function of the system is given by:

$$G_0(s) = \frac{77.8421}{0.0311s^2 - 30.5250} \tag{4.14}$$

CHAPTER 5

SIMULATION RESULTS AND THEIR INTERPRETATION

In this chapter, results were discussed for implementation of "Simulated Annealing Algorithm" on "Travelling Salesman Problem" for a specific number of cities at a time. After that, for comparison results obtained from "Tabu Search" on "Travelling Salesman Problem" were discussed.

Simulation results obtained for 10, 20, 30, 50, 75, 442, 535 cities "Travelling Salesman Problem" by "Simulated Annealing Algorithm" and 30, 50, 75, 442, 535 cities "Travelling Salesman Problem" by "Tabu Search Algorithm" is as follows:

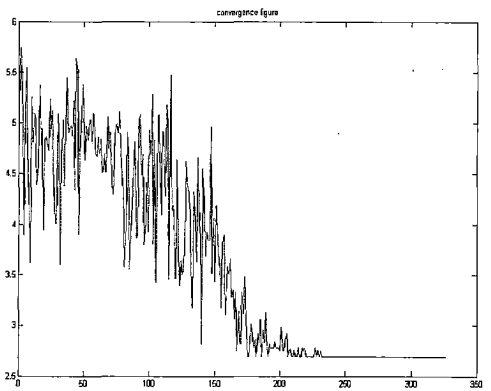

Figure 5.1: Convergence Figure obtained by "Simulated Annealing Algorithm" for 10 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

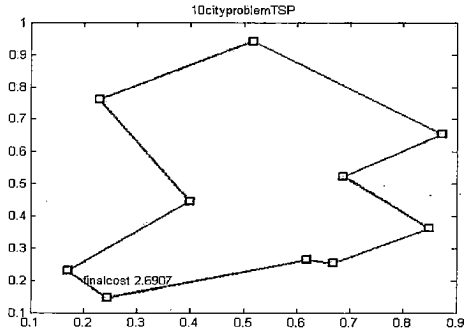

Figure 5.2: Route obtained by "Simulated Annealing Algorithm" for 10 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

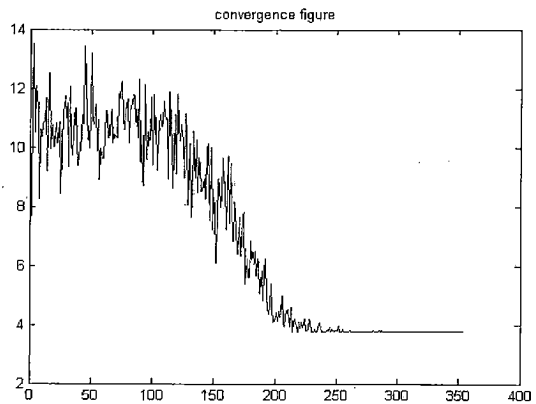

Figure 5.3: Convergence Figure obtained by "Simulated Annealing Algorithm" for 20 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

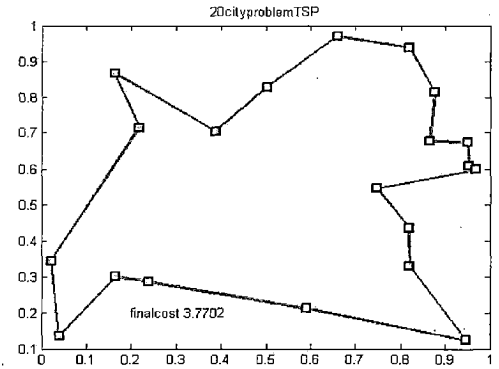

Figure 5.4: Route obtained by "Simulated Annealing Algorithm" for 20 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

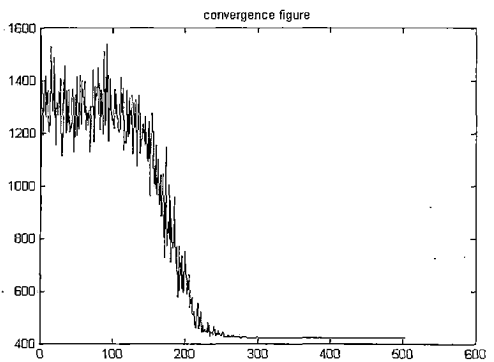

Figure 5.5: Convergence Figure obtained by "Simulated Annealing Algorithm" for 30 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

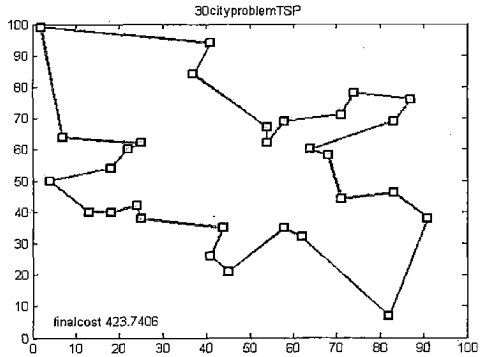

Figure 5.6: Route obtained by "Simulated Annealing Algorithm" for 30 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

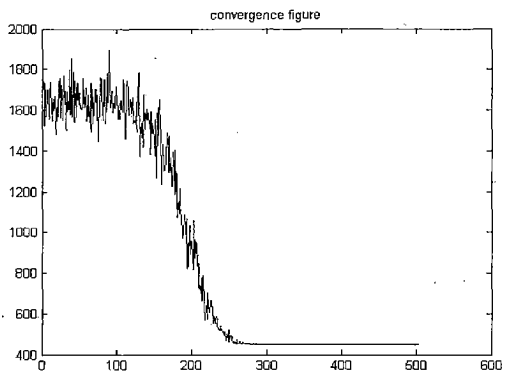

Figure 5.7: Convergence Figure obtained by "Simulated Annealing Algorithm" for 50 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

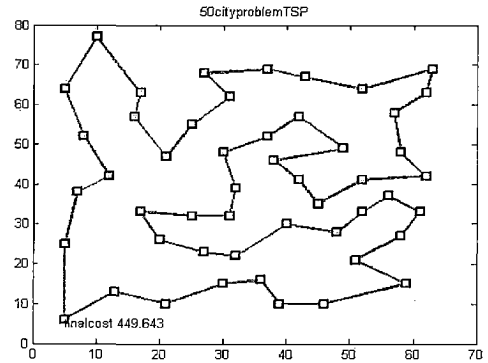

Figure 5.8: Route obtained by "Simulated Annealing Algorithm" for 50 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

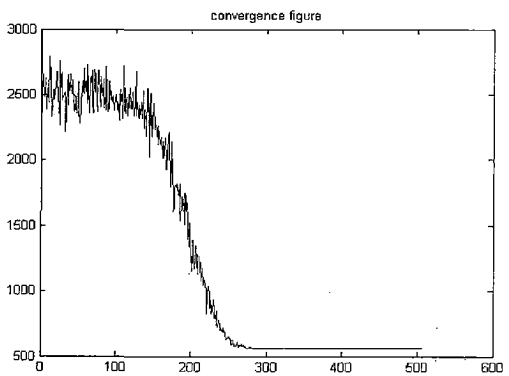

Figure 5.9: Convergence Figure obtained by "Simulated Annealing Algorithm" for 75 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

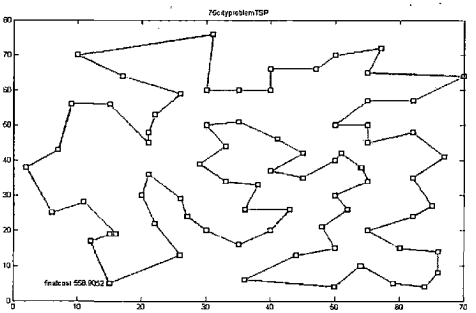

Figure 5.10: Route obtained by "Simulated Annealing Algorithm" for 75 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

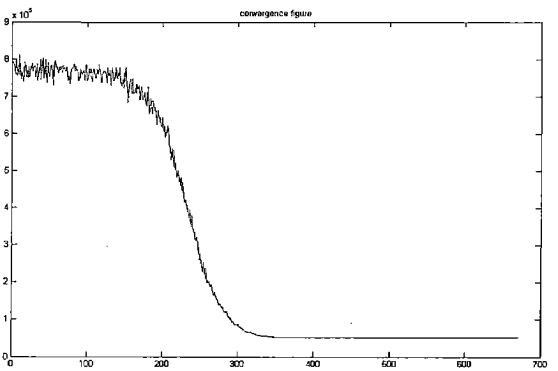

Figure 5.11: Convergence Figure obtained by "Simulated Annealing Algorithm" for 442 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

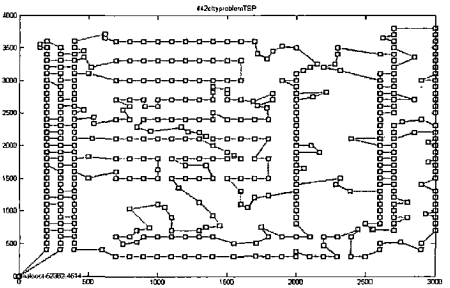

Figure 5.12: Route obtained by "Simulated Annealing Algorithm" for 442 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

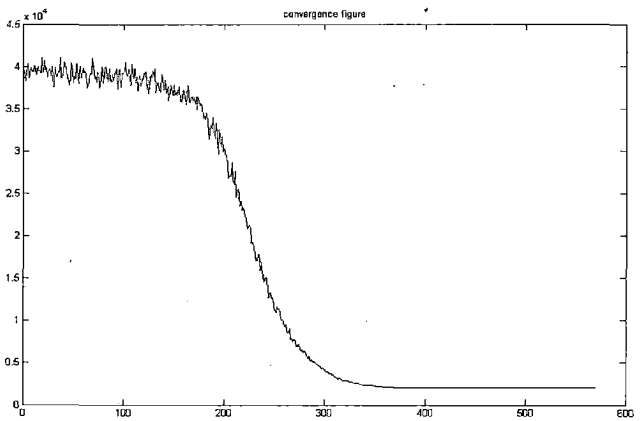

Figure 5.13: Convergence Figure obtained by "Simulated Annealing Algorithm" for 535 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost" and Cooling Rate is 0.97)

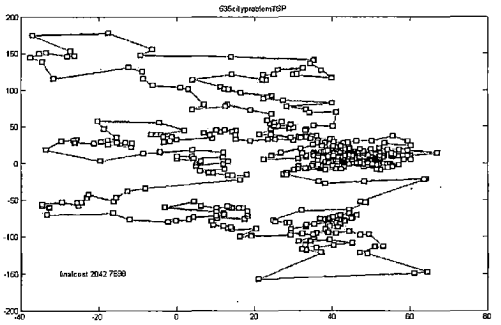

Figure 5.14: Route obtained by "Simulated Annealing Algorithm" for 535 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City and Cooling Rate is 0.97)

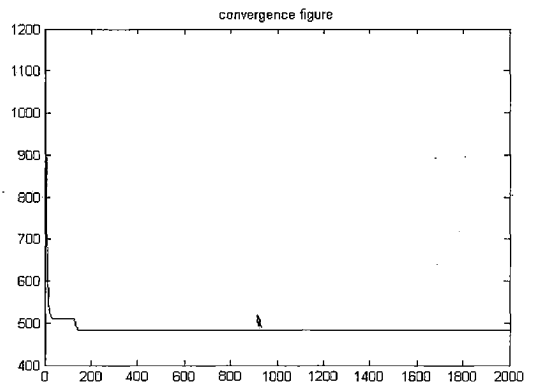

Figure 5.15: Convergence Figure obtained by "Tabu Search Algorithm" for 30 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")

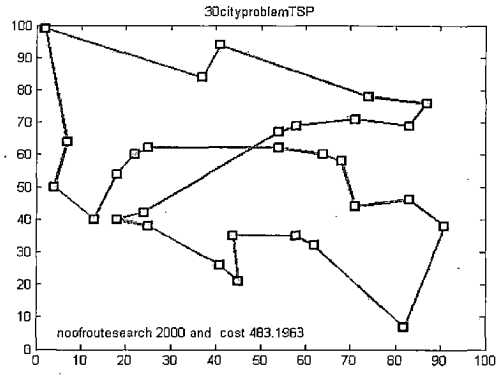

Figure 5.16: Route obtained by "Tabu Search Algorithm" for 30 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)

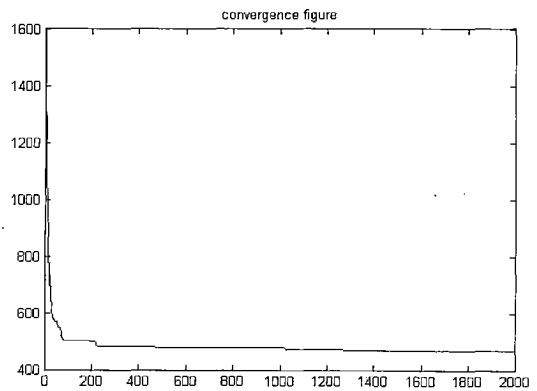

Figure 5.17: Convergence Figure obtained by "Tabu Search Algorithm" for 50 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")

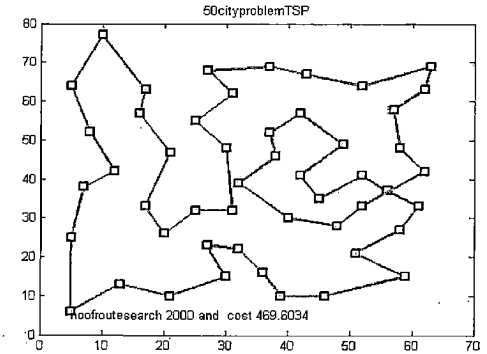

Figure 5.18: Route obtained by "Tabu Search Algorithm" for 50 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)

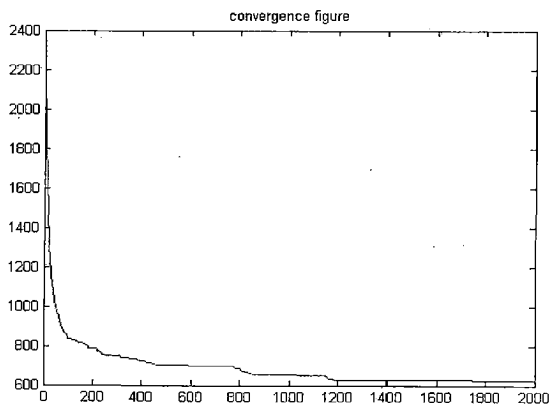

Figure 5.19: Convergence Figure obtained by "Tabu Search Algorithm" for 75 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")

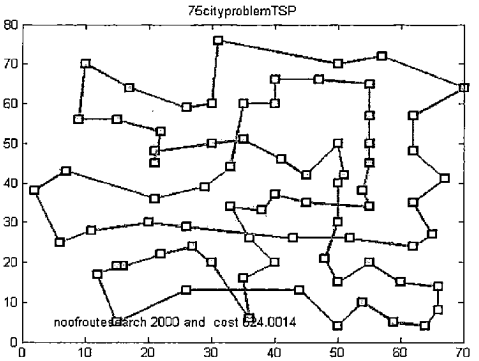

Figure 5.20: Route obtained by "Tabu Search Algorithm" for 75 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)

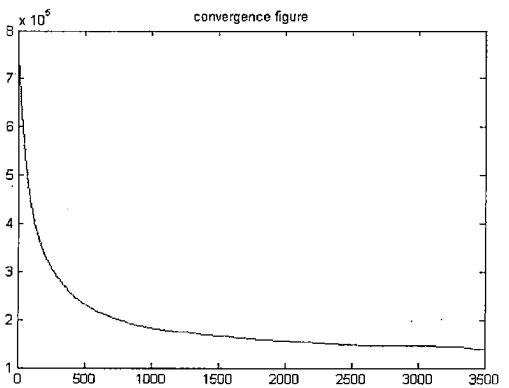

Figure 5.21: Convergence Figure obtained by "Tabu Search Algorithm" for 442 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")

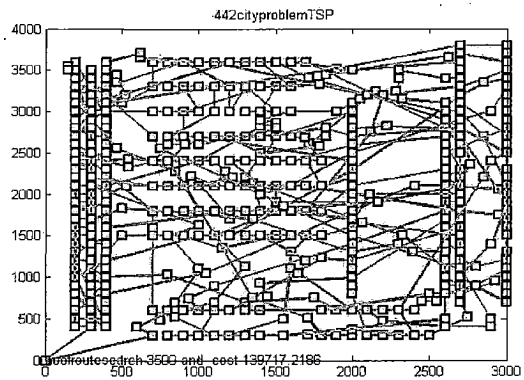

Figure 5.22: Route obtained by "Tabu Search Algorithm" for 442 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)

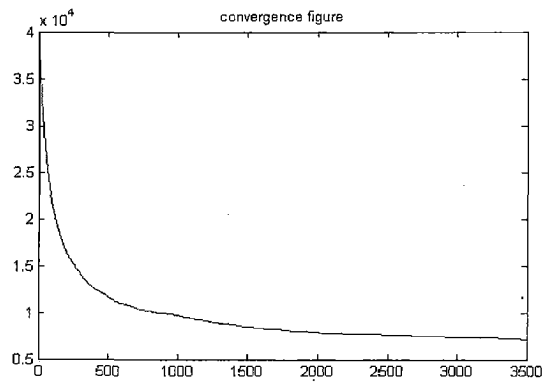

Figure 5.23: Convergence Figure obtained by "Tabu Search Algorithm" for 535 Cities Travelling Salesman Problem (On X-axis "Number of iteration" And on Y "Cost")

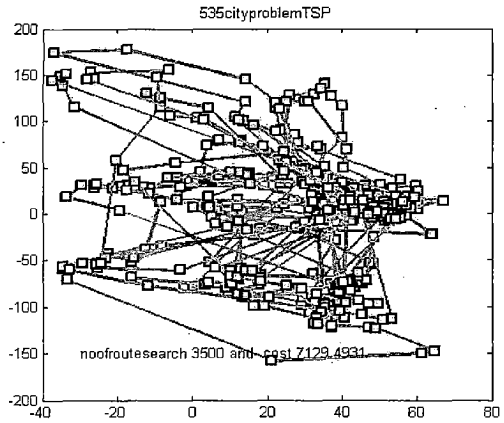

Figure 5.24: Route obtained by "Tabu Search Algorithm" for 535 Cities Travelling Salesman Problem (X-axis represent the X-coordinate of the City and Y-axis represent the Y-coordinate of the City)

Comparison of "Simulated Annealing Algorithm" and "Tabu Search Algorithm" for "Travelling Salesman Problem" [6] is given the table:

Number of Cities	Cost obtained from Simulated Annealing Algorithm	Cost obtained from Tabu Search Algorithm
30	423.7406	483.1963
50	449.643	469.6034
75	558.9072	624.0014
442	52362.4514	139717.2186
535	2042.7898	7129.4931

Table 5.1: Comparison of Cost function value obtained from SA and TS

From the above Table 5.1, there is clear idea obtained that the performance of "Simulated Annealing Algorithm" is better than "Tabu Search Algorithm" for "Travelling Salesman Problem". For large problem size of "Travelling Salesman Problem" performance of "Simulated Annealing Algorithm" is much better than "Tabu Search".

CHAPTER 6

CONCLUSION AND FUTURE PROSPECTIVE

6.1. Conclusion:

As given in chapter 1, "Travelling Salesman Problem", is a classical combinatorial optimization class of problem. In combinatorial class of problem the objective function has to be maximized or minimized according to the requirement. Here, the minimization type of problem has taken into account. In "Travelling Salesman Problem", the objective function is to minimize the roundtrip distance with the constraint that every city (represented by a point in the graph) must be travelled by a person (salesman) once and only once and return to their starting city.

Here, some assumption has been taken into consideration, that there a direct path from one city to the other cities has been existed and path is straight line path. One more consideration is taken into consideration, that all path existed in the system are "two way path".

From the previous study, a clear idea has been gained that for a large size "Travelling Salesman Problem" "Simulated Annealing Algorithm" is much more effective in finding the minimum roundtrip distance (i.e. cost) for given data than "Tabu Search Algorithm". This advantage is obtained from the fact that "Simulated Annealing Algorithm" accepts the "Uphill moves". Due to this fact, the ability of escaping from a local minimum with some probabilistic nature has been added to enhance the performance of "Simulated Annealing Algorithm" that may lead to find "Global Minima" for the given data. This kind of attribute is not present in "Tabu Search Algorithm" (i.e. it will not accepts the uphill moves), therefore, there are great chances that it will stuck in the "Bad Local Minima" and this will rarely converge toward the "Global minimization".

For very large size of "Travelling Salesman Problem" the convergence of the problem is very slow and it takes a large span of time due to extensive computation. This drawback has been

compensated with the performance obtained from "Simulated Annealing Algorithm" as compared to "Tabu Search Algorithm".

6.2. Future Prospective:

There are basically two streams for future improvement.

1. The convergence of the algorithm can be made fast by parallel implementation of the algorithm.
2. The assumption that all existed path from a city to other cities are two way and lies in the straight line can be left.

And further, these two will accommodate into a single problem.

REFERENCES

Jünger, Gerhard Reinelt and Giovanni RinaMi, "The Traveling Salesman [illegible]", chapter 4, M.O. Ball et al., Eds., Handbooks in OR & MS, VoL 7, Elsevier Science, 1995.

S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi," Optimization by Simulated Annealing ", *Science*, Volume 220, Number 4598, pp. [illegible] 680may 1983.

V. Cerny, Communicated by S. E. Dreyfus "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *Journal of Optimization Theory and* [illegible], vol. 45, no. 1, pp. 41–51, Jan. 1985.

Nicholas Metropolis, Arianna W. Rosenbluth, and Augusta H. Teller, "Equations of State calculation by Fast Computing Machines", *The Journal of Chemical Physics*, Vol. 21, No. 6, [illegible] 1953.

[illegible] and Donald Geman "Stochastic Relaxation, Gibb's Distribution and Bayesian's Restoration of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 6, Nov 1984.

Miroslaw Malek, Mohan Guruswamy, Mihir Pandya, and Howard Owens, "Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Travelling Salesman [illegible], Vol. 21, pp. 59–84, 1989.

[illegible] Monte Carlo Revolution," *Journal of Bull. Amer. Math.* [illegible] 205, 2009.

Masaya [illegible]wa, Hironori Yamauchi, and Hidekazu Terai, "Hybrid Architecture of Genetic Algorithm and Simulated Annealing," *Engineering Letters.*, 16:3, EL_16_3_11, [illegible] 2008.

[illegible] Computer," A distributed implementation of simulated annealing [illegible] travelling salesman problem", *Journal of Parallel Computing*, vol 10, Issue 3, pp.335-338, North- Holland, 1989.

Aravind Seshadri, "Simulated Annealing Algorithm for Travelling Salesman Problem", *Matlab Central.*

Lab Manual, "Inverted Pendulum" Googol Tech.

[redacted], "Ball and Beam System" Googol Tech.
[redacted], "Magnetic Levitation System" Googol Tech.