

MARKOV CHAIN MONTE CARLO
AND
THE TRAVELING SALESMAN PROBLEM

By

LIANG FA MING

A

Thesis

Submitted to

the Graduate School

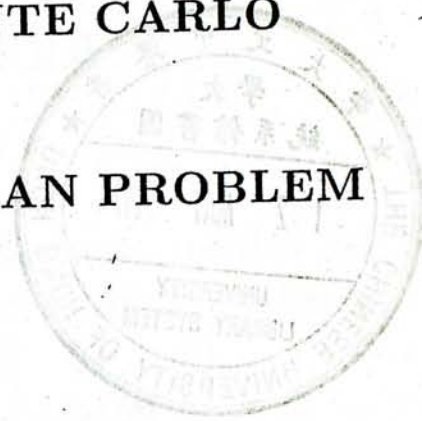
Department of Statistics

of

The Chinese University of Hong Kong

In Partial Fulfilment of
the Requirements for the Degree of
Master of Philosophy
(M.Phil.)

Oct. 1995

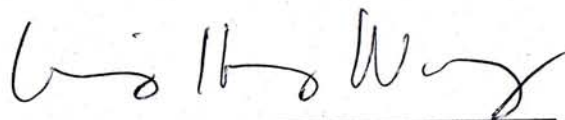




THE CHINESE UNIVERSITY OF HONG KONG

GRADUATE SCHOOL

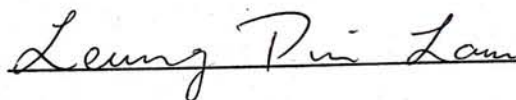
The undersigned certify that we have read the thesis, entitled "Markov Chain Monte Carlo and the Traveling Salesman Problem", submitted to the Graduate School by LIANG Fa-Ming (梁 發 明) in partial fulfillment of the requirement for the degree of Master of Philosophy in Statistics. We recommend that it be accepted.



Prof. W.H. WONG, Supervisor



Dr. K.H. LI



Dr. P.L. LEUNG

Prof. Albert, Lo

External Examiner

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

ACKNOWLEDGEMENT

The author is deeply indebted to his supervisor, Prof. Wong Wing Hung, for his generosity of encouragement and supervision. During the research program, Prof. Wong gave me numerous suggestions. I am extremely grateful to Dr. K.H. Li and Dr. P.L. Leung for their comments on an earlier version of this thesis.

CONTENTS

	page
ABSTRACT	1
CHAPTER 1 : Introduction	2
1.1 : The TSP Problem	2
1.2 : Application	3
CHAPTER 2 : Review of Exact and Approximate Algorithms for TSP	4
2.1 : Exact Algorithm	4
2.2 : Heuristic Algorithms	8
CHAPTER 3 : Markov Chain Monte Carlo Methods	16
3.1 : Markov Chain Monte Carlo	16
3.2 : Conditioning and Gibbs Sampler	17
3.3 : The Metropolis-Hasting Algorithm	18
3.4 : Auxiliary Variable Methods	21
CHAPTER 4 : Weighted Markov Chain Monte Carlo Method	24
CHAPTER 5 : Traveling Salesman Problem	31
5.1 : Buildup Order	33
5.2 : Path Construction through a Group of Points	34
5.3 : Solving TSP Using the Weighted Markov Chain Method	38
5.4 : Temperature Scheme	40
5.5 : How to Adjust the Constant Prior-Ratio	41
5.6 : Validation of Our Algorithm by a Simple Example	41
5.7 : Adding/Deleting Blockwise	42
5.8 : The sequential Optimal Method and Post Optimization	43
5.9 : Composite Algorithm	44
5.10: Numerical Comparisons and Tests	45
CHAPTER 6 : Conclusion	48
REFERENCES	49
APPENDIX A.....	54
APPENDIX B.....	58
APPENDIX C.....	61

Markov Chain Monte Carlo and the Traveling Salesman Problem

Abstract

A new Markov chain Monte Carlo method, weighted Markov chain Monte Carlo method, is suggested in this thesis. The new method is a generalization of the Metropolis algorithm and incorporates the idea of importance weighting. It is applied successfully to the traveling salesman problems. Computational experiments carried out on both randomly generated problems and problems described in the operations research literatures confirm the effectiveness of the method. Weighted Markov chain Monte Carlo method is a general method. It can be applied widely to simulation problems involving multimodal surfaces and to simulation and optimization problems with buildup structures.

Chapter 1

Introduction

1.1 The TSP Problem

Let N be the number of cities and $D = (d_{ij})$ be the distance(cost) matrix whose elements $d_{ij} (\geq 0)$ denote the distance between city i and city j . The *traveling salesman problem (TSP)* is to find a permutation π of the cities that minimizes the quantity

$$\sum_{i=1}^{N-1} d_{\pi(i), \pi(i+1)} + d_{\pi(N), \pi(1)}.$$

This quantity is referred to as the tour length, since it is the length of the tour a salesman would make if he starts some city, visits the other cities in the order specified by the permutation, and then returns to the city from which he starts.

If the distance matrix is symmetric, i.e. $d_{ij} = d_{ji}$, for $1 \leq i, j \leq N$, we say the problem is *symmetric*; otherwise, it is *asymmetric*. Also, D is said to satisfy the triangle inequality if, and only if $d_{ij} + d_{jk} \geq d_{ik}$ for $1 \leq i, j, k \leq N$. This occurs in *Euclidean* problems, i.e. when the cities are in \mathcal{R}^2 and d_{ij} is the straight-line distance between city i and city j .

There are $(N-1)!/2$ feasible solutions for N -city TSP. It is impossible to exhaustively search the whole solution space in a large scale problem. TSP is a NP-complete (non-deterministic polynomial time complete) problem. No method for exact solution with a computing effort bounded by a power of N has been found. However, a number of special cases of TSP are solvable in polynomial time (see, for example, Gilmore, Lawler and Shmoys, 1985). Examples of such problems include the TSPs where $D = (d_{ij})$ is an

upper triangular matrix, i.e. $d_{ij} = 0$ for all $i \geq j$ and a class of job sequencing problems defined by Gilmore and Gomory(1964).

1.2 Application

The traveling salesman problem has many applications (Lawler et al. 1985, Laporte 1992). Here we just give some examples.

1. *Computer wiring*(Lenstra and Rinnooy Kan 1975)

Some computer components can be described as modules with pins attached to them. It is often desired to link these pins by means of wires, so that exactly two wires are attached to each pin. In order to avoid signal cross-talk and to improve ease and neatness of wirability, the total wire length has to be minimized.

2. *Physical mapping*(Cuticchia et al. 1992)

In constructing physical mapping based on hybridisation fingerprinting data, if the penalty function is defined to be the sum of pairwise discrepancies between adjacent objects(probes or clones), then the problem can be formulated as a TSP .

3. *Job sequencing*

Suppose n jobs must be performed sequentially on a single machine and that d_{ij} is the change-over time if job j is executed immediately after job i . The optimal ordering of the jobs is obtained by solving a TSP.

In chapter 2 we will give a review of exact and approximate algorithm for TSP. In chapter 3 we will review the construction methods of Markov chain. In chapter 4 the weighted Markov chain method is proposed. In chapter 5 the weighted Markov chain method is applied to the traveling salesman problem, computational experiments carried out on both randomly generated problems and problems described in the operations research literature. Chapter 6 concludes with a discussion.

Chapter 2

Review of Exact and Approximate Algorithms for TSP

2.1 Exact Algorithms

A large number of exact algorithms have been proposed for TSP based on integer linear programming (ILP). In this section we give a number of ILP formulations and the algorithms derived from these formulations.

2.1.1 Integer linear programming formulations

One of the earliest formulations is due to Dantzig, Fulkerson and Johnson (DFJ)(1954). It associates one binary variable x_{ij} to every arc (i, j) , equal to 1 if, and only if (i, j) is used in the optimal solution, $i \neq j$. For the sake of simplicity, let V be the set of all cities(points). The DFJ formulation is

$$\text{Minimize } \sum_{i \neq j} d_{ij} x_{ij} \quad (2.1)$$

subject to

$$\sum_{j=1}^N x_{ij} = 1, \quad i = 1, \dots, N, \quad (2.2)$$

$$\sum_{i=1}^N x_{ij} = 1, \quad j = 1, \dots, N, \quad (2.3)$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, \dots, N, \quad i \neq j, \quad (2.4)$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad \text{for any } S \subset V, \quad 2 \leq |S| \leq N - 2. \quad (2.5)$$

Where $|S|$ denotes the number of arcs contained in set S . In this formula, the objective function clearly describes the cost of the optimal tour. Constraints (2.2) and (2.3) specify that every vertex is entered exactly once and left exactly once, respectively. Constraint (2.4) imposes binary conditions on the variables. Constraints (2.5) are subtour-breaking constraints; they prohibit the formation of subtours, i.e. tours on subsets of less than N vertices. Although the subtour-breaking constraints can be formulated in many different ways, the one given above is very intuitive. If there is a subtour on some subset S of V , this subtour will contain $|S|$ arcs. Consequently, the left-hand side of the inequality will be equal to $|S|$, which is larger than $|S| - 1$, and the constraint is violated for this particular subset. Without the subset-breaking constraints, the TSP reduces to an assignment problem(AP).

A number of alternative formulations have been proposed by many authors, but none of them seems to have a stronger linear relaxation than DSJ(Langevin, Soumis and Desrosiers, 1990).

2.1.2 Branch and Bound algorithm

Branch-and-Bound(BB) algorithm is a well-known enumerative search technique for obtaining optimal solution to asymmetric TSP(Lawler et al. 1985). The feasible solution are divided into a collection of disjoint sets. Lower bounds are obtained for each set, and are used in conjunction with upper bounds so that some sets are discarded from further consideration. The procedure is applied recursively to each of the sets. The recursion explodes exponentially whenever the lower bounds are weak with respect to the optimal solution value. The AP-relaxation is used to generate lower bounds on the optimum value. So the quality of a BB algorithm is directly related to the quality of the bound provided by the relaxation. Many authors have proposed Branch and Bound algorithms for the TSP based on AP relaxation. The following algorithm is proposed by Carpaneto and Toth(1980), and is a good example of this class of algorithms.

In their method, the problem solved at a generic node of the search tree is a modified assignment problem (i.e. x_{ii} is fixed at 0 for all i) in which some x_{ij} variables are fixed at 0 or 1. If the AP solution consists of a unique tour over all vertices, it is then feasible for the TSP. Otherwise, it consists of a number of subtours. One of these subtours is selected and broken by creating subproblems in which all arcs of the subtour are in turn prohibited. We will use the following notations:

z^* : the cost of the best TSP solution so far identified;

z_h : the value of the objective function of the modified AP at node h of the search tree;

\underline{z}_h : a lower bound for z_h ;

I_h : the set of included arcs (x_{ij} variables fixed at 1) at node h of the search tree;

E_h : the set of excluded arcs (x_{ij} variables fixed at 0) at node h of the search tree.

step 1(Initialization): Obtain a first value for z^* by means of a suitable heuristic. Create node 1 of the search tree: set $I_1 = E_1 = \emptyset$, and obtain z_1 by solving the associated modified AP. If $z_1 \geq z^*$, stop: the heuristic solution is optimal. If the solution contains no illegal subtours, it constitutes the optimal tour: stop. Otherwise, insert node 1 in a queue.

step 2(Node selection): If the queue is empty, stop. Otherwise, select the next node (node h) from the queue. Here we use a *breadth first* rule, i.e. branching is always done on the pendant node having the lowest z_h .

step 3(Subproblem partitioning): The solution obtained at node h is illegal and must be broken by partitioning the current subproblem into descendant subproblems h_r characterized by sets I_{h_r} and E_{h_r} . In order to create these subproblems, consider a subtour having the least number s of arcs not belonging to I_h . Let these arcs be $(i_1, j_1), \dots, (i_s, j_s)$, in the order in which they appear in the subtours. Then create s subproblems with

$$I_{h_r} = \begin{cases} I_h & r = 1 \\ I_h \cup \{(i_u, j_u) : u = 1, \dots, r-1\} & r = 2, \dots, s \\ E_{h_r} = E_h \cup \{(i_r, j_r)\}, & r = 1, \dots, s. \end{cases}$$

Execute steps 4-6 for $r = 1, \dots, s$.

step 4(Bounding): Compute a lower bound \underline{z}_{h_r} on z_{h_r} by row and column reduction of the cost matrix. If $\underline{z}_{h_r} < z^*$, proceed to step 5. Otherwise, consider the next r and repeat step 4.

step 5(Subproblem solution): Solve the subproblem associated with node h_r (a modified AP restricted by I_{h_r} and E_{h_r}). If $z_{h_r} \geq z^*$, consider the next r and proceed to step 4.

step 6(Feasibility check): If $z_{h_r} \leq z^*$, check whether the current solution contains subtours. If it does, insert node h_r in the queue. Otherwise, set $z^* = z_{h_r}$ and store the tour, if $z^* = z_h$, go to step 2.

Using their algorithm, the authors have consistently solved randomly generated 240-vertex TSPs in less than one minute on a CDC6600. The main limitation of this algorithm appears to be computer memory rather than CPU time.

Recently, Miller and Pekny (1991) have proposed a new powerful BB algorithm for large *asymmetric* TSP. Consider the dual AP(DAP):

$$\begin{aligned} \text{Maximize} \quad & \sum_{i=1}^n u_i + \sum_{j=1}^n v_j \\ & d_{ij} - u_i - v_j \geq 0, \quad i, j = 1, \dots, N, i \neq j \end{aligned}$$

Denote by $z^*(\text{TSP})$ the optimal TSP solution value, by $z^*(\text{AP})$ the optimal value of the AP linear relaxation, and by $z^*(\text{DAP})$ the optimal value of the dual AP linear relaxation. Clearly $z^*(\text{AP}) = z^*(\text{DAP})$. Moreover, note that $z^*(\text{AP}) + (d_{ij} - u_i - v_j)$ is a lower bound on the cost of an AP solution that includes arc (i, j) . Miller and Pekny make use of this in an algorithm that initially removes from consideration all x_{ij} variables whose cost d_{ij} exceeds a threshold value λ . Consider a modified problem TSP' with associated linear assignment relaxation AP' and its dual DAP', obtained by redefining the costs d_{ij} as follows:

$$d_{ij}' = \begin{cases} d_{ij} & \text{if } d_{ij} \leq \lambda \\ \infty & \text{otherwise} \end{cases}.$$

The authors prove the following proposition which they use as a basis for their algorithm: an optimal solution for TSP' is optimal for TSP if

$$z^*(\text{TSP}) - z^*(\text{AP}) \leq \lambda + 1 - u_i' - v_{\max}' \quad (2.6)$$

and

$$\lambda + 1 - u_i' - v_{\max}' \geq 0 \quad (2.7)$$

for $i = 1, \dots, n$, where u' and v' are optimal solution to DAP', and v_{\max}' is the maximum element of v' . The quantity $\lambda + 1 - u_i' - v_{\max}'$ underestimates the smallest reduced cost of any discarded variable. The algorithm can be summarized as follows:

- 1) Choose λ .
- 2) Construct (d_{ij}') and solve TSP'.
- 3) If (2.6) and (2.7) hold, then $z^*(TSP') = z^*(TSP)$: stop. Otherwise, double λ and goto step 2.

The authors report that if λ is suitably chosen in step 1 (e.g., the largest arc cost in a heuristic solution), there is rarely any need to perform a second iteration. To solve TSP', the authors have developed a BB algorithm based on the AP relaxation. They have applied this procedure to randomly generated asymmetric problems. Instances involving up to 5000 vertices were solved within 40 seconds on a Sun 4/330 computer. The largest problem reported solved by this approach contains 500000 points and required 12623 seconds of computing time on a Cray 2 supercomputer. But applied to symmetric problems, the method was able to solve only small problems. They applied the algorithm to the 532-city problem of Padberg and Rinaldi (1987), and only find a solution within 8.2% of optimal in 18 seconds and within 6.8% of optimal in 130 seconds on a Sun 4/330 workstation.

There are many other methods proposed for solving TSP exactly, see Laporte(1992) for a review in depth.

2.2 Heuristic Algorithms

TSP is a NP-hard problem even with additional constraints such as triangle inequality or Euclidean distances. It is impossible to find its exact solution for a large scale problem within reasonable amount of computation time. Accordingly, heuristic algorithms are often preferred to exact algorithm for solving the large TSPs that occur in practice (e.g.,

drilling problems). These have motivated many authors to design the heuristic algorithms for the TSP. A good heuristic algorithm should have the following properties(Ong and Huang, 1989):

- 1) It should yield a solution within a reasonable amount of computation time.
- 2) The solution should be close to the optimum, on average.
- 3) The probability of producing a solution which deviates substantially from the optimum should be small.

Ong and Huang(1989) had studied asymptotic expected performance of some TSP heuristics based on their empirical evaluation.

Generally speaking, TSP heuristics can be classified as tour construction procedures, tour improvement procedures, and composite procedures, which base on both construction and improvement techniques.

2.2.1 Construction Procedures

Construction procedures usually build up a tour gradually by selecting each vertex in turn and then inserting them one by one into the current tour. Various criteria have been proposed to select the next vertex to insert and to identify the best place to insert. These include the proximity to the current tour and the minimum detour. The most popular tour construction heuristics include savings heuristic(Clarke and Wright, 1963), cheapest insert heuristic and nearest neighbor heuristic. Rosenkrantz et al. (1977) has compared the performances of these methods.

2.2.2 Improvement Procedures

If one is not satisfied with a tour constructed by the heuristics, one may attempt to improve it by local searching (Aarts and Korst 1989) based on stepwise improvement on the value of the cost function by exploring neighbourhoods. The fundamental concept, *neighbourhood structure*, is defined as follows:

T' is a neighbour solution of T if and only if T' can be obtained from T by a legitimate perturbation.

The tour improvement algorithm based on such a structure consists of two subroutines: (A) Initialization which is used to construct an initial tour T_0 , and (B) Improvement which is used to determine if there is neighbouring tour of better cost, and if so will return one such tour T' . If no such tour exists, T is called "locally optimal". These subroutines may be either randomized or deterministic. Thus a local optimization algorithm has the following structure:

- 1) Initialization: construct an initial solution $T = T_0$.
- 2) Improvement: do the following until T is declared locally optimal,
 - 2.1) generate T' from the neighborhoods of T .
 - 2.2) If T' is a better solution than T , set $T = T'$.
- 3) Stop, and return T .

The local algorithms stop at a local optimum, which need not be globally optimal, and there is generally no guarantee as to how far this local optimum is from a global optimum (Johnson, Papadimitriou and Yannakakis, 1985). The quality of the solution obtained by a local search algorithm usually strongly depends on the initial solution ; and for most combinatorial optimization problems no guideline are available for an appropriate choice of the initial solution (Aarts and Korst 1989). The following are some examples of the algorithms:

1 r-opt algorithm

The most famous tour improvement methods for TSP are the "2-opt", "3-opt" and "Lin-Kernighan". The corresponding neighbourhood structures are as follows:

2-opt (Lin, 1965): Two tours are neighbours if one can be obtained from the other by deleting two edges, reversing one of the resulting two paths, and reconnecting. Typically, it can remove the cross lines among 4 cities (The move to such a neighbour is generally known as a Lin-move).

3-opt (Lin, 1965): Two tours are neighbours if one can be obtained from the other by

deleting three edges and putting the three resulting paths together in a new way, possibly reversing one or more of them.

Lin-Kernighan (Lin and Kernighan 1973) : The neighbourhood structure is too complicated to be described here in detail. In summary, for a tour T' to be a neighbor of tour T , it must be shorter and there must be a way to obtain it by breaking k (k is not prefixed) edges of T , rearranging them, and then performing a series of additional 2-opt moves.

Recently Johnson(1990) has developed a randomized iterated Lin-Kernighan method that produces near-optimal solutions.

2 Simulated Annealing

The idea of Markov chain construction for Monte Carlo simulation was introduced in (Metropolis et al. 1953). Kirkpatrick et al. (1983) adapted this approach for discrete global optimization incorporating the concept of the annealing process in thermodynamics. Annealing refers to a process used to reveal the low temperature state of some material. The material is first melted and then slowly cooled with long time spent at successive lower temperatures near the freezing point. The time spent at each temperature must be sufficiently long to allow a thermal equilibrium to be realized. Simulated annealing means simulating the annealing process by a Monte Carlo method (random changes in the state of the system), where the global minimum of the objective function represents the low energy configuration. The algorithm can be summarized as following:

- 1) Initialization. Obtain an initial solution $s = s_0$ and temperature $t = t_0$.
- 2) Improvement. Repeat the following until its time to quit:
 - 2.1) choose a random neighbour s' of s .
 - 2.2) If $c(s') < c(s)$ set $s = s'$.
 - 2.3) otherwise,
 - 2.3.1) choose a random number r uniformly from $[0,1]$.
 - 2.3.2) if $r \leq \exp(-(c(s') - c(s))/t)$, set $s = s'$.
 - 2.4) determine a new temperature t according to some preselected cooling schedule.
- 3) Return s (or best solution seen so far, if different).

An important characteristic of this algorithm is that sometimes the next state accepted may have higher energy than the previous one. This reduces the probability of becoming trapped in a local optimum. The acceptability of uphill moves is affected by a control parameter t called the temperature. Typically the temperature is gradually reduced from a high value, at which most uphill moves are accepted, to a low one at which few if any such moves are accepted. One standard scheme of theoretical interest is "logarithm" cooling, in which the temperature on the k th trial is set to $C/\log(k)$ for some fixed constant C . This cooling schedule will normally guarantee convergence to an optimal solution (Geman and Geman 1984) (although the convergence time is likely to exceed the time needed to find an optimal solution by exhaustive search). In Stander and Silverman (1994), a dynamic programming approach is used to find the temperature schedule which is optimal for a simple minimization problem, and the temperature schedule is compared with other non-standard choices, including straight, geometric and reciprocal. A more commonly used scheme is geometric cooling, in which the temperature is fixed for some prespecified constant L number of trials, and is then multiplied by some prespecified reduction factor, which is a constant smaller than but close to 1, such as 0.95 or 0.9. This provides no guarantees for convergence, but seems to work well in practice. Even geometric cooling, however, leads to substantially greater running times (for acceptable results) than needed by the corresponding local optimization scheme.

Simulated Annealing has been applied to TSP by many authors including Bonomi and Lutton (1984), Rossier, Troyon and Liebling (1986), Golden and Skiscim (1986) and Nahar, Sahni and Shragowitz (1989), with apparently a mixed degree of success.

3 *Tabu Search* (Glover 1989, 1990, 1993; Glover and McMillan, 1986)

As in the previous two methods, successive neighbours of a solution s are examined and, as for simulated annealing, the objective is allowed to deteriorate in order to avoid local minima. In order to prevent cycling, solutions that have already been examined are forbidden and inserted in a constantly updated 'tabu list'. The method can be summarized as the following:

- 1) Initialize to obtain an initial solution s . Set the tabu list $T = \emptyset$.
- 2) Let $N(s)$ be a neighbour of x . If $N(s) \setminus T = \emptyset$, go to step 3. Otherwise, identify a least cost solution s' in $N(s) \setminus T$ and set $s = s'$. Update T and the best known solution.
- 3) If the maximum number of allowed iterations since the beginning of the process or since the last update has been reached, stop. Otherwise, go to step 2.

The success of this method depends on the careful choice of a number of control parameters. Fiechter (1990) applied tabu search to the TSP with seemingly very positive results.

4 Genetic Algorithm

The genetic algorithm is different from the general tour improvement methods based on iterative local search, such as *r-opt*, *simulated annealing* and *tabu search*. These methods find a solution of which the quality depends on its initial solution. Sometimes in order to obtain a better quality solution, multiple independent runs are needed. No use is made in latter runs of the results of earlier runs. But the genetic algorithm (Holland, 1975) capitalizes on those earlier results. It first performs some fixed number of independent runs (possible in parallel), and then derives new starting solutions based on a transfer of information between (a mating of) the solutions found. This process can then be repeated for many generations until progress stops being made, with the population at each mating step consisting of the newly constructed tours and a selection of the best ones from previous generations. It can be summarized as follows:

- 1) Set $t = 0$, initialize to obtain initial population $P(t)$, and then evaluate structures in $P(t)$.
- 2) While termination condition is not satisfied, do
 - 2.0) set $t = t + 1$.
 - 2.1) select mating populating $M(t)$ from $P(t - 1)$.
 - 2.2) recombine structures in $M(t)$.
 - 2.3) evaluate structures in $M(t)$.
 - 2.4) replace some or all of $P(t - 1)$ with $M(t)$ to form $P(t)$.

3) Return best solution

The algorithm was first applied to TSP by Brady(1985), who mated two tours by looking for a pair of subpaths, one in each tour, that contain precisely the same set of cities. The longer of the two paths is then replaced in its tour by the shorter. The local optimization algorithm used was 2-opt. Unfortunately the results for a 64-city geometric example were no better than performing multiple independent runs of 2-Opt for the same time.

There are many variations in the selection of the mating population and the recombination of the structures. The choices of them determine the efficiency of the algorithm. In order to search from as large a solution space as possible to obtain a better quality solutions, a mutation process is usually executed after the recombination of the structures.

2.2.3 Composite Algorithm

In recent years, two effective composite algorithm have been developed. The first is the CCAO heuristic(Goldean and Stewart, 1985) . The second is GENIUS (Gendreau, Hertz and Laporte, 1992).

1. CCAO algorithm

This heuristic is designed for symmetrical Euclidean TSPs. It exploits a well-known property of such problems, namely that in any optimal solution, vertices located on the convex hull of all vertices are visited in the order in which they appear on the convex hull boundary(Flood, 1956). The method can be summarized as follows:

step 1(C: convex hull). Define an initial(partial) tour by forming the convex hull of vertices.

step 2(C: cheapest insertion). For each vertex k not yet contained in the tour, identify the two adjacent vertices i_k and j_k on the tour such that $c_{i_k,k} + c_{k,j_k} - c_{i_k,j_k}$ is minimized.

step 3(A: largest angle). Select the vertex k^* that maximizes the angle between edges (i_{k^*}, k^*) and (k^*, j_{k^*}) on the tour, and insert it between i_{k^*} and j_{k^*} .

step 4 Repeat step 2 and 3 until a Hamiltonian tour of all vertices is obtained.

step 5(O: Or-opt). Apply the r-opt procedure to the tour and stop.

The rationale behind steps 2 and 3 is that by selecting k^* so as to maximize the angle it makes with the tour, the solution remains as close as possible to the initial convex hull.

2. The GENIUS algorithm

One major drawback of the CCAO algorithm is that its insertions are executed sequentially without much concern for global optimality. They may result in bad decisions that the post-optimization phase will be unable to undo. GENIUS executes each insertion more carefully, by performing a limited number of local transformations of the tour, simultaneously with the insertion itself. It contains two parts: a generalized insertion phase(GENI) and a post-optimization phase (US) that successively removes (unstring) vertices from the tour and reinserts (string) them, using the generalized insertion rule. The algorithm can be summarized as follows:

step 1 Create an initial tour by selecting an arbitrary subsets of three vertices. Initialize the p -neighbourhoods of all vertices.

step 2 Arbitrarily select a vertex v not yet on the tour. Implement the least cost insertion of v by considering the two possible orientations of the tour and the two insertion types. Update the p -neighbourhoods of all vertices to account for the fact that v is now on the tour. If all vertices are on tour, go to step 3. Otherwise repeat this step .

step 3 Consider the tour s with cost z obtained above. Set $s^* = s$, $z^* = z$, and $t = 1$.

step 4 Apply the unstringing and stringing procedures with vertex v_t , and consider in each case the two possible types of operation and the two-tour orientations. Let s' be the tour obtained and z' be its cost. Set $s = s'$, and $z = z'$.

4.1) if $z < z^*$, set $s^* = s$, $z^* = z$ and $t = 1$; repeat step 4.

4.2) if $z \geq z^*$, set $t = t + 1$.

4.3) if $t = n + 1$, stop: the best available tour is s^* and its cost is z^* . Otherwise go to step 4.

Chapter 3

Markov Chain Monte Carlo Methods

3.1 Markov Chain Monte Carlo

To demonstrate the idea of Markov chain Monte Carlo instead of a rigorous mathematical discussion, we will mainly use the elementary language and concepts corresponding to discrete state spaces. Extensive theoretical accounts can be found in Besag and Green(1993) and Tierney(1994).

Suppose that we wish to generate a sample from a distribution $\pi(x)$ for $x \in \mathcal{X} \subseteq \mathcal{R}^n$ but cannot do this directly. However, if we can construct a Markov chain with state space \mathcal{X} and equilibrium distribution $\pi(x)$. Then we run the chain which can be simulated directly for a long period of time, and the simulated values of the chain can be used as a basis for inference.

Clearly, successive samples of the chain X^t will be correlated. Therefore, more samples are needed than would be required if independent samples are possible.

Under suitable regularity conditions, the realization $X^1, X^2, \dots, X^n, \dots$ from an appropriate chain, has the following asymptotic results:

$$X^t \xrightarrow{d} X \sim \pi(x) \text{ as } t \rightarrow \infty$$
$$\frac{1}{t} \sum_{i=1}^t f(X^i) \rightarrow E_{\pi} f(X) \text{ a.s. as } t \rightarrow \infty$$

If the first asymptotic result is to be exploited to mimic a random sample from $\pi(x)$, suitable spacings will be required between realizations used to form the sample, or parallel independent runs of the chain might be considered. The second asymptotic result implies that ergodic averaging of a function of interest over realizations from a single run of the chain provides a consistent estimator of its expectation, so the samples of the chain can be used for Monte Carlo integration.

To implement this strategy, we simply need algorithms for constructing chains with specified equilibrium distributions. We now consider some particular forms of Markov chain schemes.

3.2 Conditioning and Gibbs Sampler

Let $\pi(x) = \pi(x_1, \dots, x_k)$, $x \in \mathcal{R}^k$, denote a joint density, and let $\pi(x_i|x_{-i})$ denote the induced full conditional densities for each of components x_i , given values of the other components $x_{-i} = (x_j, j \neq i)$, $i = 1, \dots, k$, $1 < k \leq n$.

A systematic form of the so-called Gibbs sampler algorithm (Geman and Geman, 1984) proceeds as follows.

- 1) Pick arbitrary starting values $x^0 = (x_1^0, \dots, x_k^0)$.
- 2) Successively make random drawings from the full conditional distributions $\pi(x_i|x_{-i})$, $i = 1, \dots, k$, as follows:

$$\begin{aligned} & x_1^1 \text{ from } \pi(x_1|x_{-1}^0); \\ & x_2^1 \text{ from } \pi(x_2|x_1^1, x_3^0, \dots, x_k^0); \\ & x_3^1 \text{ from } \pi(x_3|x_1^1, x_2^1, x_4^0, \dots, x_k^0); \\ & \vdots \\ & x_k^1 \text{ from } \pi(x_k|x_{-k}^1). \end{aligned}$$

This completes a transition from $x^0 = (x_1^0, \dots, x_k^0)$ to $x^1 = (x_1^1, \dots, x_k^1)$. Iteration of this process generates a sequence $x^0, x^1, \dots, x^t, \dots$ which is a realization of a Markov chain, with equilibrium distribution $\pi(x)$ and transition probability from x^t to x^{t+1} given

by

$$K_G(x^t, x^{t+1}) = \prod_{l=1}^k \pi(x_l^{t+1} | x_j^t, j > l, x_j^{t+1}, j < l)$$

The key feature of this algorithm is that we only sample from the full conditional distributions $\pi(x_i | x_{-i})$.

Another algorithm which involves successive drawing from various (not just full) conditionals is the substitution sampler, which is discussed in detail in Gelfand and Smith (1990), who examine its relation to the Gibbs sampler and to data augmentation (Tanner and Wong 1987). A detail description of this method, including developments and applications can be found in Tanner (1991).

3.3 The Metropolis-Hasting Algorithm

To construct a Markov chain $X^1, X^2, \dots, X^t, \dots$ with state space \mathcal{X} and equilibrium distribution $\pi(x)$, the Metropolis-Hastings algorithm constructs the transition probability from $X^t = x$ to the next realized state X^{t+1} as follows.

Let $q(x, x')$ denote a transition probability function (essentially can be any distribution), such that, if $X^t = x$, x' drawn from $q(x, x')$ is considered as a proposed possible value for X^{t+1} .

Then we calculate the ratio

$$\alpha(x, x') = \begin{cases} \min\left\{\frac{\pi(x')q(x',x)}{\pi(x)q(x,x')}, 1\right\} & \text{if } \pi(x)q(x, x') > 0 \\ 1 & \text{if } \pi(x)q(x, x') = 0. \end{cases}$$

We actually accept $X^{t+1} = x'$ with probability $\alpha(x, x')$; otherwise, we reject the proposed transition and set $X^{t+1} = x$.

This construction defines a Markov chain with transition kernel function given by

$$p(x, x') = \begin{cases} q(x, x')\alpha(x, x') & \text{if } x' \neq x \\ 1 - \sum_{x''} q(x, x'')\alpha(x, x'') & \text{if } x' = x. \end{cases}$$

It is easy to check the transition kernel satisfies the detailed balance, i.e.

$$\pi(x)p(x, x') = \pi(x')p(x', x).$$

Which guarantees that $\pi(x)$ is the equilibrium distribution of the constructed chain.

This general algorithm is due to Hasting(1970); see, also, Peskun(1973).

Clearly, different specific choices of $q(x, x')$ will lead to different specific algorithms. Tierney(1994) provides a systematic taxonomy of the kinds of choice available. Here we just follow him to give some simple examples.

1) Metropolis algorithm

If $q(x, x') = q(x', x)$, we have $\alpha(x, x') = \min\{\pi(x')/\pi(x), 1\}$, which is the well-known Metropolis algorithm (Metropolis et al 1953), which forms the basis of the simulated annealing method (Kirkpatrick et al 1983).

2) Random walk Chain

If $q(x, x') = q'(x' - x)$, the chain is driven by a random walk process. Natural choices for the increment distribution include a uniform distribution on a disk, a normal distribution or perhaps a multivariate t -distribution. Split- t distribution(Geweke, 1989) may also be useful.

3) Independence chains

If $q(x, x') = q'(x')$, and the acceptance probability $\alpha(x, x')$ can be written as

$$\alpha(x, x') = \min\left\{\frac{w(x')}{w(x)}, 1\right\}$$

where $w(x) = \pi(x)/q(x)$. This function w is the importance weight function that would be used in importance sampling if observations were generated from the density $q(x)$.

The independence Metropolis chain is closely related to the corresponding importance sampling process. Candidate steps with low weights are rarely accepted. On the other hand, candidates with high weights are usually accepted, and the process will usually remain at these points for several steps, thus using repetition to build up weight on these points within the samples. If some points have very high weight values, then the process may *get stuck* at these points for a long time. Insight for the general importance sampling method, multivariate t -distribution with low degrees of freedom, split t -distributions and other distributions may be good choices for proposal function $q(x')$.

4) Rejection Sampling Chain

Rejection sampling(Ripley 1987) is the basis for several algorithms for generating vari-

ates from standard univariate distributions.

Suppose we have a density h and a constant c such that, hopefully, $\pi(x)/h(x) \leq c$ for all x . The sampling procedure is as follows:

- (1) Generate Z from h .
- (2) generate u from uniform $(0,1)$.
- (3) If $u \leq \pi(Z)/ch(Z)$, accept Z ; otherwise, repeat step (1)-(3).

The final Z has density

$$q(x) \propto \pi(x) \wedge ch(x).$$

If $\pi(x) \leq ch(x)$ is true, then q is proportional to π and the Z s are i.i.d samples from π . But usually it is very difficult to ensure that c is large enough for $ch(\cdot)$ to dominate π without choosing c excessively large, leading to an inefficient algorithm with many rejections. And even then we still can not be certain that $ch(\cdot)$ does dominate $\pi(\cdot)$ without extensive analysis of the tails of h and π .

Using this rejection scheme to drive an independent Metropolis chain provides a simple remedy. If we define $C = \{x : \pi(x) \leq ch(x)\}$, then the Metropolis acceptance probability can be written as

$$\alpha(x, x') = \begin{cases} 1 & \text{for } x \in C \\ \frac{ch(x)}{\pi(x)} & \text{for } x \notin C, x' \in C \\ \min\left\{\frac{\pi(x')h(x)}{\pi(x)h(x')}, 1\right\} & \text{for } x \notin C, x' \notin C. \end{cases}$$

Thus the algorithm will occasionally reject some proposed transitions when the chain is at a point $x \notin C$. This repeats compensates for the deficiency in the envelope at x . In the sampling process the dependence between samples is introduced to make up for this deficiency and leaves π invariant for the Markov chain.

5) Combining strategies

The methods described above can be used separately, or they can be combined into hybrid strategies. Mainly there are 3 strategies to combine them.

- 1) Using an Markov chain Monte Carlo method within the conditioning of the Gibbs sampler.

2) Mixtures. Suppose positive probabilities $\alpha_1, \dots, \alpha_m$ are specified, and at each step one of the kernels is selected according to these probabilities.

3) Cycles. Each kernel is used in turn, and when the last one is used the cycle is restarted. (for detail, see Tierney, 1994).

3.4 Auxiliary Variable Methods

In the method of auxiliary variables, the state variable x is augmented by one or more additional variables $u \in U$; in some contexts, u may have a physical interpretation in the original process, but often it is quite abstract. The joint distribution of x and u will be defined by taking the given distribution of interest $\pi(x)$ as the marginal for x , and specifying the conditional $\pi(u|x)$; this can be chosen quite arbitrarily. So we have

$$\pi(x|u) \propto \pi(x, u)$$

We now construct a Markov chain on $\mathcal{X} \times U$ that alternates between two types of transition:

- 1) u is drawn from $\pi(u|x)$;
- 2) x' is generated given u and x .

Note that the transition kernel function $P(x \rightarrow x'; u)$ should preserve detailed balance for the conditional $\pi(x|u)$, i.e.

$$\pi(x|u)P(x \rightarrow x'; u) = \pi(x'|u)P(x' \rightarrow x; u) \quad (3.1)$$

The simplest example of such a transition function is

$$P(x \rightarrow x'; u) = \pi(x'|u) \quad (3.2)$$

for which the resulting method amounts to the Gibbs sampler applied blockwise to x and u in turn, but there are many other choices that can be made. If we define

$$P(x \rightarrow x') = \sum_u \pi(u|x)P(x \rightarrow x'; u)$$

This is just the double transition function from x to x' . Since

$$\pi(x) \sum_u \pi(u|x)P(x \rightarrow x'; u) = \sum_u \pi(u)\pi(x|u)P(x \rightarrow x'; u)$$

and

$$\pi(x') \sum_u \pi(u|x') P(x' \rightarrow x; u) = \sum_u \pi(u) \pi(x'|u) P(x' \rightarrow x; u)$$

From (8), we have

$$\pi(x) P(x \rightarrow x') = \pi(x') P(x' \rightarrow x)$$

This leaves $\pi(x)$ invariant for the chain, so such an approach construct a valid Markov chain Monte Carlo procedure for $\pi(x)$, provided that irreducibility and aperiodicity can be demonstrated; for the case of equation (3.2) it is clearly sufficient that there exists u^* such that $\pi(u^*|x)$ is positive for all x .

The most successful auxiliary variables method has been the Swendsen and Wang(1987) algorithm, using equation (3.2) (the $1/T_k$ power of $\pi(\cdot)$), designed specifically for the Potts model, the multicolor generalization of Ising model. The algorithm provides a remarkably simple means with which to combat the problems of critical slowing-down, encountered by single-component updating.

A development with wider implications has been the introduction of *simulated tempering* into the literature on auxiliary processes(Marinari and Parisi, 1992; Geyer and Thompson, 1995). It speeds up the mixing rate of the whole system, by devising an ordered sequence of chains, which, at one extreme, has the target distribution(with lower temperature) as its limit, at the other extreme, is a rapidly mixing chain(with higher temperature). The chains can be run in parallel, with intermittent proposals to swap states of adjacent chains and with acceptance probability that ensure each maintains its own limiting distribution. The occurrence of swaps can substantially increase mixing. In Geyer and Thompson (1995) only one chain is running at any particular time. Moves from the current chain to the adjacent chain(s) are proposed periodically, again such that the individual limit distributions are maintained. Thus, at one extreme, information can be collected on the target chain. In applying the method to ancestral inference, Geyer and Thompson (1995) created the sequence of densities by setting the penetrances to be various convex combinations of two basic sets of values. One corresponds to the genetic model of interest, the other corresponds to a model that is easy to simulate. But sometimes it is very difficult to move from one chain to its adjacent ones, especially when there

are much difference between two adjacent chains. For a detail description of this method, see Besag and Green (1993).

Chapter 4

Weighted Markov Chain Monte Carlo Method

Suppose we want to draw samples from a density function

$$P(x_1, x_2, \dots, x_n) = P(X) = cf(X)$$

where c is the normalizing constant, and $f(X)$ is a function that can be evaluated numerically. Metropolis et al.(1953) introduced the following method:

Given the current sample X , draw X^* according to a distribution $T(X \rightarrow X^*)$, where T can be essentially any distribution. It is called the proposal function. Define the ratio

$$r = r(X \rightarrow X^*) = \frac{f(X^*)T(X^* \rightarrow X)}{f(X)T(X \rightarrow X^*)}$$

and sample a variable u from the uniform distribution $U(0, 1)$. If $u \leq r$ set $X' = X^*$ else $X' = X$. This define a Markov transition $x \rightarrow x'$. The corresponding transition kernel function $K(X \rightarrow X')$ satisfies the detailed balance:

$$f(X)K(X \rightarrow X^*) = f(X^*)K(X^* \rightarrow X)$$

Hence such a transition will leave the density $f(\cdot)$ invariant.

Suppose we have obtained samples X_1, X_2, \dots, X_n using this transition. Let $\phi_n(x)$ be the distribution of X_n , the sample obtained in n th step. Then $\phi_n(x)$ have the property that asymptotically

$$\lim_{n \rightarrow \infty} \phi_n(x) = P(x)$$

If the function $f(x)$ has multiple modes, and there are deep and wide platforms between modes, it is impossible to simulate the function $f(x)$ correctly by this method. The reason can be derived directly from detailed balance which implies that the Markov chain will find it difficult to leave a mode for the platform region and hence it has little chance to jump to another mode.

We now give an example to demonstrate this. Let $f(x)$ be a linear combination of two normal distributions:

$$f(x) = \frac{1}{4}N(5, 0.1) + \frac{3}{4}N(20, 0.1)$$

Note that it has two widely separated modes. We run the Metropolis algorithm 20 times independently, with 50,000 trial steps per run. The starting value is set to be $(0, 0)$ in each run. The proposal function is $N(0, 0.1)$. The simulated function $\hat{f}(x)$ is shown in figure 1.

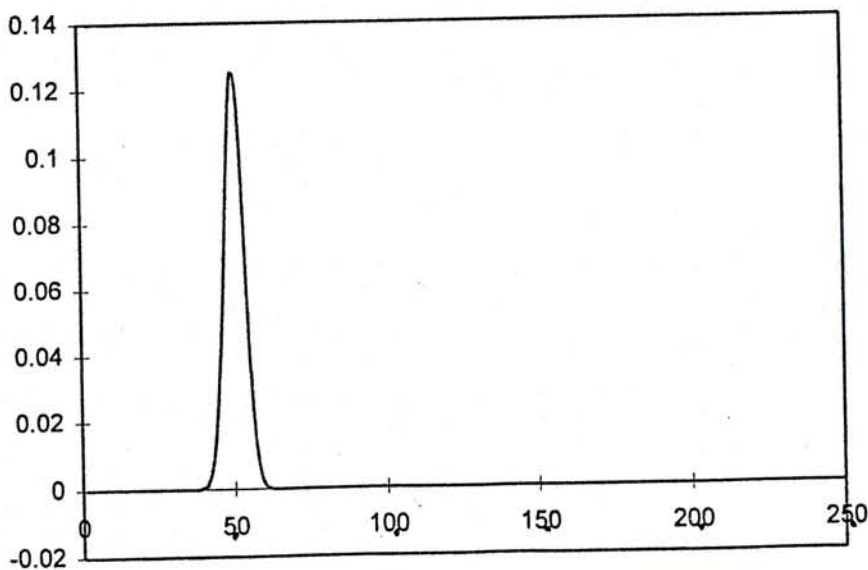


figure 1

The simulation result for $f(x)$ using Metropolis algorithm

It is clear that the Markov chain is always trapped in the first mode. Several proposal functions, including $N(0, 0.1)$, $U(0, 0.1)$, t -distribution with low degree of freedom have been tried. But all of them can only simulate one mode, and the estimated function is completely wrong.

A potential remedy that has sometimes been suggested is to use several or many

different runs, starting from different points, scattered around the parameter space. As an exploratory strategy, this may be quite informative, particularly if the modes can be used as starting points. In practice, the main modes will often be deduced separately by deterministic hill climbing. However, the problem of how to combine the separate runs into coherent inferences then arises. Ideally, one would like each run to be sufficiently long so that it could sample all the modes frequently. In that case we have almost the correct long run proportions, and multiple runs have no intrinsic merit.

Now we modify the algorithm by introducing the auxiliary variable w as the weight of the state x , so now the state space is $\mathcal{X} \times \mathcal{W}$ instead of \mathcal{X} , and the current state is (x, w) , instead of x . Let $T(X \rightarrow x^*)$ be the proposal function and $\theta = \theta(x, w)$. The algorithm can be summarized as follows:

1) Draw x^* according to $T(x \rightarrow x^*)$.

2) Calculate

$$r = R(x, w; x^*) = w \frac{f(x^*)T(x^* \rightarrow x)}{f(x)T(x \rightarrow x^*)} \quad (4.1)$$

3) If $r \geq \theta$, accept the proposed transition, and set $(x', w') = (x^*, r)$,

else generate a random variable $U_1 \sim U(0, 1)$

a) if $U_1 \leq r/\theta$, accept the proposal transition and set $(x', w') = (x^*, \theta)$.

b) if $U_1 > r/\theta$, set $(x', w') = (x, w/q)$.

Where q is the rejection probability of (x, w) ,

$$q = q(x, w) = \int_{r < \theta} \left(1 - \frac{R(x, w; x^*)}{\theta}\right) T(x \rightarrow x') dx'.$$

4) Let (x', w') be the current state, go to step 1).

An alternative choice for b) in the above pseudocode of the algorithm is to keep drawing new proposal states until we find an acceptable one. Then new state is set as (x', w') ,

where

$$w' = \begin{cases} r * p & \text{if } r \geq \theta \\ \theta * p & \text{otherwise,} \end{cases}$$

$$p = 1 - q(x, w) = 1 - \int_{r < \theta} \left(1 - \frac{r}{\theta}\right) T(x \rightarrow x') dx'.$$

For x^* with a value r larger than θ , it is assigned a weight larger than θ . This makes it easy to leave the current state for a new state. For x^* with a value of r smaller than θ , if it is rejected, one choice is to return state $(x, \frac{w}{q})$, where $\frac{w}{q} > w$, this makes the chain easier to leave the current state for a new state. Another choice is to keep drawing new proposal states until we find an acceptable one. We usually take this second choice when the weight is larger than a given upper control value. This algorithm substantially speeds up the rate of mixing of the Markov chain. The following theorem shows that after integrating out the variable w , we can get the correct distribution which is just what we want to sample from.

THEOREM: If the following equality holds,

$$\int g(x, w)w dw = cf(x).$$

Let

$$h(x', w') = \int \int g(x, w)k((x, w) \rightarrow (x', w'))dw dx,$$

where $k((x, w) \rightarrow (x', w'))$ is the transition kernel function. Then

$$\int h(x', w')w' dw' = cf(x')$$

(proof can be found in Appendix A)

Since both choices leave the function $cf(x)$ invariant, we can alternatively choose one in every step of the simulation process.

In the algorithm, θ is a prefixed function of (x, w) . Note that given

$$\int g(x, w)w dw \propto f(x),$$

we always have

$$\int \int \int g(x, w)k((x, w) \rightarrow (x', w'))dx dw dw' \propto f(x').$$

This does not depend on the form of the function $\theta(\cdot, \cdot)$. On the other hand, the acceptance probability for every proposal transition, $\min\{r/\theta, 1\}$, depends on θ , so we can tune θ to get appropriate moving probability for every state according to its current weight w . Let

$$\theta = w \left(1 - \frac{w}{M+w}\right) \left(1 - \frac{1}{M + \frac{1}{w}}\right)$$

Where M is a prefixed constant, and it is the upper control value for w . The plot θ vs w for fixed M ($M = 20,000$) is in figure 2.

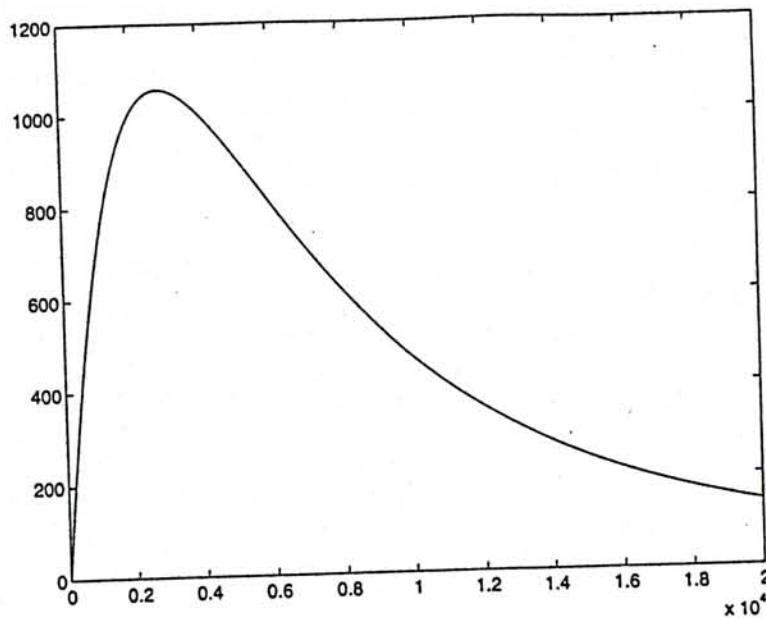


figure 2

The plot θ vs w for fixed $M = 20,000$

In figure 2, when w increases from a very small value, the corresponding θ increases very quickly, so then w can be adjusted quickly to be a large value by many rejections. With increasing w , θ will be increasing up to a maximum value, after that it is decreasing and eventually asymptotes to 1. The decreasing of θ results in fewer rejections, which implies that the proposed transition with the small ratio r can be accepted and then w is set to a small value. This prevents the value of θ from becoming too large. Such a choice is found to be helpful in controlling the magnitudes of the weights.

In the simulation process of the Markov chain, $\frac{1}{q}$ and p need to be estimated for setting the new weight during rejection steps. How to estimate them ?

The simplest method is that we draw n samples according to the transition density $T(x \rightarrow x')$ if the proposed transition is rejected. We know that

$$p = \int_{r \geq \theta} T(x \rightarrow x') dx' + \int_{r < \theta} \frac{r}{\theta} T(x \rightarrow x') dx'$$

We can estimate p by

$$\hat{p} = \frac{1}{n} \sum_{i=1}^n (I(r_i \geq \theta) + \frac{r_i}{\theta} I(r_i < \theta)) = \frac{1}{n} \sum_{i=1}^n p_i$$

where

$$p_i = \min(1, \frac{r_i}{\theta})$$

Clearly, \hat{p} is an unbiased estimate of p .

An unbiased estimate of p^2 is

$$\hat{p}^2 = \frac{1}{n(n-1)} [(\sum_{i=1}^n p_i)^2 - \sum_{i=1}^n p_i^2].$$

To estimate $\frac{1}{q}$, first note that

$$\frac{1}{q} = \frac{1}{1-p} = 1 + p + p^2 + \dots$$

When p (e.g. $\hat{p} \leq 0.1$) is small, $1 + \hat{p} + \hat{p}^2$ is a good estimator of $\frac{1}{q}$, although it is a biased estimator.

When p is large (e.g. $\hat{p} > 0.1$), the estimator of $\frac{1}{q}$ with small bias based on Taylor expansion usually includes high order terms of p . In this case we usually use $\frac{1}{1-\hat{p}}$ to estimate $\frac{1}{q}$ directly instead of basing on Taylor expansion.

To save computation time in the second choice of rejection step, we usually check the previous n samples used to estimate p again to try to accept one of them, e.g. the sample x' , the k th sample, is accepted, then x_k , k should satisfy the following equality

$$k = \min\{j : r_j \geq \theta \text{ or } U_j < \frac{r_j}{\theta}, j = 1, 2, 3, \dots, n\}.$$

Where U_j is drawn from uniform distribution $U(0, 1)$, but if

$$\{j : r_j \geq \theta \text{ or } U_j < \frac{r_j}{\theta}, j = 1, 2, 3, \dots, n\} = \emptyset$$

we continue to sample x' according to function $T(x \rightarrow x')$ until we find an acceptable proposal transition. The new samples can then be used to get a better estimator of p together with the previous n samples.

Using this algorithm, we simulate the function

$$f(x) = \frac{1}{4}N(5, 0.1) + \frac{3}{4}N(20, 0.1)$$

using $N(0, 0.1)$ as the proposal function, and the weighted Markov chain is run 20 times independently with 50,000 trial steps per run, the starting point is set to be $(0, 0)$ in each run. The simulated function \hat{f} is shown in figure 3.

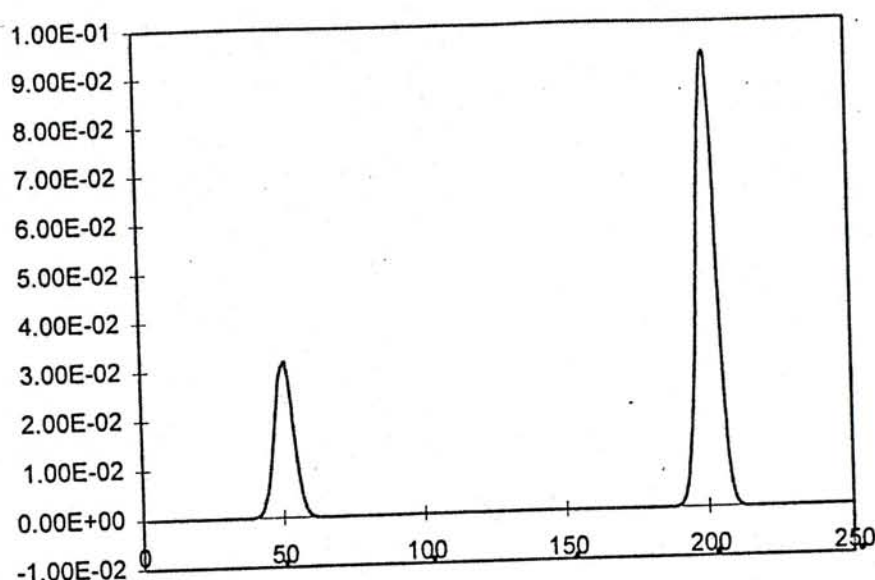


figure 3

The simulation result for $f(x)$ using weighted Markov chain

Clearly we obtain a correct simulated distribution. The algorithm correctly locates the two modes. The ratio of the two modes is estimated correctly, too. So then we can say that the Markov chain has mixed well. The algorithm works equally well for other proposal functions, such as $U(0, 0.1)$, Cauchy and t -distribution with low degree of freedom.

Note that the initial value of the weight is not very important, because the weights can be adjusted automatically in the simulation process.

In our algorithm, if w is fixed to a constant (arbitrary except 0), and let θ be the same constant as w , then the algorithm is just the Metropolis algorithm (w and θ is fixed to 1, Metropolis et al 1953). Clearly, the Metropolis algorithm satisfies our theorem. This can be verified easily. We can thus regard the Metropolis algorithm as a special case of our algorithm.

Chapter 5

Traveling Salesman Problem

Based on the deep similarity between statistical mechanics and combinatorial optimization, we can use the Boltzmann distribution [Toda, Kubo and Saitô, 1983] to characterize the tour. This distribution gives the probability of a permutation I of vertices with length l_I at temperature T , and is given by

$$P(X = I) = \frac{1}{Z(T)} \exp\left(-\frac{l_I}{K_B T}\right) \quad (5.1)$$

where X is a stochastic variable denoting the permutation of points, $Z(T)$ is the partition function, which is defined as

$$Z(T) = \sum_J \exp\left(-\frac{l_J}{K_B T}\right),$$

in which K_B is the Boltzmann constant, \sum denotes a sum over all possible permutations.

Note that the global maximum point of the function $P(X)$ is just the solution of TSP. Instead of attacking TSP directly, we solve the related problem of how to efficiently sample from the Boltzmann distribution at a fixed low temperature.

The popular algorithm of Simulated Annealing is also based on the Boltzmann distribution. It often encounters difficulty in finding the optimal solution, because its search space is too local. It does not directly deal with the kernel of NP-problem: the exhaustive search space.

Our method to handle this difficulty is to project the high dimensional solution space into a lower dimensional space based on the structure of the problem to solve an easier

problem which has captured the most important characteristics of the original problem. In the TSP, the lower dimensional problem is also a TSP but with a small number of cities. Then cities are added into the lower dimensional tours one by one to obtain the solution of the original problem in the high dimensional space. This process is simulated by constructing a weighted Markov chain. The optimal value or near optimal value can be found according to the samples of the Boltzmann distribution.

Our approach was first outlined in Wong (1995). It has certain similarity to the method of simulated tempering recently proposed by Marinari & Parisi (1992) and Geyer & Thompson (1995). In simulated tempering, to sample from $f(\cdot)$, Marinari and Parisi (1992) propose to create a Markov chain with an augmented state vector (k, z) , where z takes values in Z and k ranges from 1 to m . In our method, we also augment the state space by an auxiliary variable. So the state is augmented to be (k, x_k) instead of being x , where k is auxiliary variable. For different k , the sample spaces for x_k need not be the same. The joint distribution for (k, x_k) is required to be proportional to $\alpha_k g(x_k|k)$, where $g(\cdot|m)$ is assumed to give the same density as $f(\cdot)$, but, for k less than m , $g(\cdot|k)$ will give densities on different spaces. This can be driven by various Markov chain construction scheme, such as the weighted Markov chain method.

The above scheme is very general and now we explain when and how this generality can be put to good use. For example, suppose after suitable parameterization, z can be written as $z = (z_1, z_2, \dots, z_n)$, and the information used to determine the density of z can be partitioned correspondingly as $y = (y_1, y_2, \dots, y_n)$. It is assumed that, based on the partial information $u_j = (y_1, y_2, \dots, y_j)$, we have a way to specify an unnormalized density $g(x_j|u_j)$ for $x_j = (z_1, z_2, \dots, z_j)$. It is required that $g(z|u_n) = f(z)$ and that, for all j , $g(x_j|u_j)$ has reasonable overlap with the marginal density of x_j under the joint density $g(x_{j+1}|u_{j+1})$. We will say that such a problem has a *sequential buildup* structure. Typically we have

$$\dim(x_k) > \dim(x_{k-1}) > \dots > \dim(x_1)$$

So the Markov chain construction involves two moves: *extrapolation* $T(x_i \rightarrow x_{i+1})$ and *projection* $T(x_i \rightarrow x_{i-1})$. Note there is no need for $g(x_1|u_1)$ to be close to the marginal of x_1 under $f(\cdot)$, although that would be an ideal situation. The method should work under

the much weaker requirement stated above.

The method involves several important considerations:

- 1) how to construct the buildup structure.
- 2) choice of $p(i, x_i)$.
- 3) choice of α_i .
- 4) choice of extrapolation and projection

With regard to (1), we note that many problems have such a structure, for example, complex missing data pattern in Gaussian models, nonparametric Bayesian analysis of binary data (see Kong, Liu and Wong 1994) and Multiloci genetic linkage analysis (Irwin, Cox and Kong 1994).

In the context of TSP, $z = (z_1, z_2, \dots, z_n)$ can be seen as the n ordered cities (the method for obtaining the order is described in the next section), and $x_j = (z_1, \dots, z_j)$, $j = 1, \dots, n$ is the tour of first j cities. So $f(x_n)$ is just the distribution we wanted for the tour length. This seems to be a natural buildup structure for the TSP.

Thus, for TSP, we have

$$p(i, x_i) = \alpha_i C_i^{-1} e^{-\frac{L_i}{t}}$$

where C_i is the normalized constant of distribution of x_i . So we should make

$$\alpha_i \propto C_i = \frac{1}{\text{normalizing factor}}$$

The remaining thing is to choose appropriate and reasonable extrapolation and projection procedures which are just our proposal functions. We will discuss them in the following sections.

5.1 Buildup Order

Based on the idea of projecting the high dimensional solution space into a lower dimensional solution space, the problem solved in the lower dimensional space should be as similar to the original one as possible.

For the traveling salesman problem, the cities to be added early should be as far away for each other as possible, i.e. if the first point is in one corner, then the next point should be in its diagonal corner. One hopes that a lower dimensional problem with only

20 percents of the cities will capture the outline of the optimal route. Based on this idea, many methods can be created to obtain the order. Let

V : set of n cities, $V = \{0, 1, 2, \dots, n - 1\}$

A : set of cities having been ordered

A^c : set of cities not yet been ordered, and $A \cup A^c = V$

d_{ij} : the distance between city i and city j

One order is given as follows,

$$\{i : \max_{j \neq i} (\frac{1}{\bar{x}_i} + \frac{1}{\min(d_{ij})}), i, j \in A^c\}$$

where $\bar{x}_i = \frac{1}{|A^c|} \sum_{j \in A^c} d_{ij}$.

Another order can be given as follows:

the next point k to add satisfies the following condition,

$$\exists m \in A, d_{km} = \max_{i \in A^c} \min_{j \in A} d_{ij}, k \in A^c$$

If there are several points satisfying the condition above simultaneously, we only select one arbitrarily.

Now we set $A = A + \{k\}$, $A^c = A^c - \{k\}$. Note that the first element of A can be selected arbitrarily. In our algorithm, the scale of the lowest dimensional problem, i.e., the points in the initial group is usually larger, e.g. 15 points. These points would have almost covered all corners of the whole tour, so our algorithm is not sensitive to the selection of the starting point. For convenience, we usually choose the point nearest to the original point or the first random point.

5.2 Path Construction Through a Group of Points

Suppose point k will be added into the current tour. The first thing to do is to find the nearest neighbour group of k in the current tour. This is just like drawing a large enough circle around point k such that the circle contains a prefixed number of points. We call these points the neighbours of point k , and call the prefixed number the size of the neighbour group.

According to the number of the paths through the neighbour group of point k , the neighbours can be classified into three types (see figure 4):

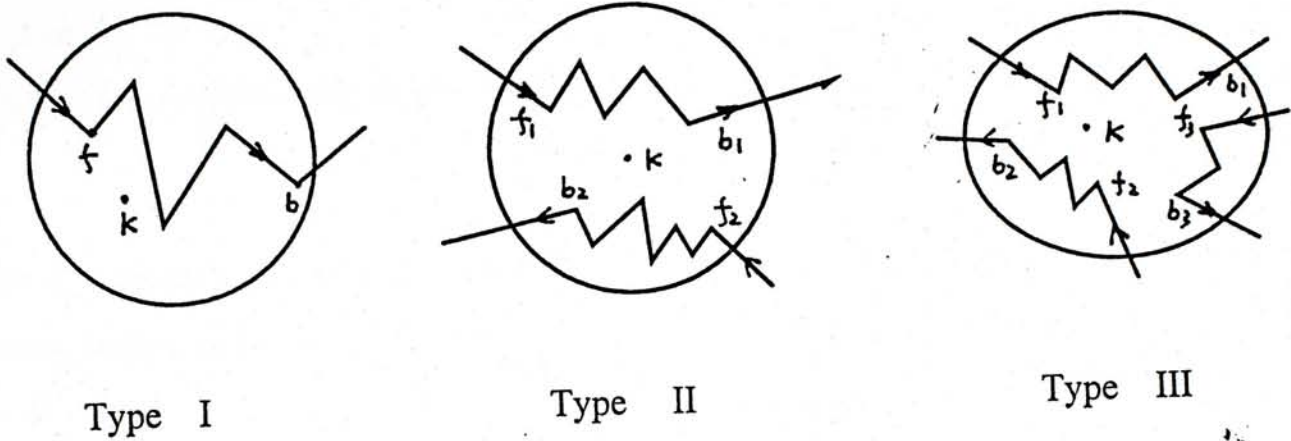


figure 4

The neighbour group of point k

For type *III*, there may be three or more paths through the neighbour group, for the sake of simplicity, we only demonstrate the type in the case of three paths. The points not on the paths, f_1 to b_1 or f_2 to b_2 , are deleted from the neighbour group. So after this reduction, type *III* can be regarded as type *II*, and the number of points in the neighbour group may be less than the prefixed neighbour group size. Note that the two paths, f_1 to b_1 and f_2 to b_2 , are just the paths which have the two smallest values of d_l among all paths which pass through the neighbour group.

Where

$$d_l = \min_{j \in \text{path}l} d_{kj} \quad l = 1, 2, 3, \dots$$

How is point k added into the current tour?

For type *I* case, the current path from entry point f to exit point b is forgotten. A new path from entry point f to exit point b , through all interior points of the neighbour group and point k , will be reconstructed. The path can be constructed as follows,

Let $G = \{ \text{all points in the neighbour group} \}$

$S = \{ i: \text{point } i \text{ has been sampled into new path, } i \in G \}$

$S^c = G - S$

$|S^c|$ = number of points in set S^c .

1) set initial value, $S = \{f, b\}$, $S^c = G - \{f, b\}$, and m , a prefixed number, usually $m = 4$ or 5 .

2) If $|S^c| \leq m$ go to step 4, else sample one point k_1 from S^c according to probability

$$\frac{\exp(-d_{f,k_1}/T)}{\sum_{j \in S^c} \exp(-d_{f,j}/T)}$$

where $d_{.,.}$ denotes the distance between two points. T is a temperature, higher than the true temperature at which we wish to simulate. We call T trial temperature. Then $S \leftarrow S + \{k_1\}$, $S^c = G - S$.

3) If $|S^c| \leq m$, go to step 4, else sample another point k_i from S^c according to probability

$$\frac{\exp(-d_{k_{i-1},k_i}/T)}{\sum_{j \in S^c} \exp(-d_{k_{i-1},j}/T)}$$

until $|S^c| \leq m$.

4) Sample one path through the remaining points in set S^c from all $m!$ permutations of m points according to the probability

$$\frac{\exp(-l_i/T)}{\sum_{j=1}^{m!} \exp(-l_j/T)}$$

where l_j denotes the total length from point k_i to b for permutation j .

Now a new path through point k and all points in the neighbour group of point k has been constructed. Its trial density is denoted by

$$q(x) = \frac{\exp(-l(x)/T)}{\Pi(x, T)}$$

where $l(x)$ is the total length of the path from f to b , through all the points in the neighbour group, and

$$\Pi(x, T) = \sum_{j \in S^c} \exp(-d_{f,j}/T) \sum_{j \in S^c - \{k_1\}} \exp(-d_{k_1,j}/T) \cdots \sum_{j=1}^{m!} \exp(-l_j/T)$$

For type II neighbour group, points b_1 and f_2 can be considered as one point, i.e., let the distance between point b_1 and f_2 equal to zero, and then it can be regarded as type I case to reconstruct a path from point f_1 to point b_2 , through point k and all interior points of the neighbour group.

Suppose now n paths (x_1, x_2, \dots, x_n) through one group of points have been sampled using the method described above. Based on the idea of importance sampling, one path can be resampled from the n paths at the true temperature. So then a shorter path can be expected to result from this resampling. The framework of resampling step is as follows,

$$x_1, x_2, \dots, x_n \stackrel{i.i.d}{\sim} q(x)$$

1) calculate a weight for each of them

$$w(x_i) = \frac{f(x_i)}{q(x_i)} = \frac{\exp(-l(x_i)/t)}{q(x_i)}$$

where t is true temperature. We repeat that usually t is smaller than the trial temperature T .

2) resample path y accord to probability

$$\frac{w(y)}{\sum_{j=1}^n w(x_j)} \quad y \in \{x_1, x_2, \dots, x_n\}$$

3) the probability of path y being chosen is

$$\begin{aligned} p^*(y) &= \sum_{i=1}^n p(x_i \text{ is chosen} | x_i = y) p(x_i = y) \\ &= np(x_1 \text{ is chosen} | x_1 = y) p(x_1 = y) \\ &= nq(y) E_{T_{n-1}} \frac{w_1(y)}{w_1(y) + T_{n-1}} \\ &= n \exp(-l(y)/t) E_{T_{n-1}} \frac{1}{w_1(y) + T_{n-1}}. \end{aligned}$$

Where

$$T_{n-1} = \sum_{j=2}^n w(x_j).$$

This gives rise to the problem of how to estimate $E_{T_{n-1}} \frac{1}{w_1(y) + T_{n-1}}$. Of course the simplest method is to sample m T_{n-1} ($T_{n-1}^1, T_{n-1}^2, \dots, T_{n-1}^m$), then use

$$\hat{E}_y = \frac{1}{m} \sum_{j=1}^m \frac{1}{w_1(y) + T_{n-1}^j}$$

to estimate it. This method is wasteful as it requires extra samples for T_{n-1} . We can assume that the neighbour group of point k has the same entry point f and exit point b

at different states (L_k, k, w) (i.e. L_k and w may be different). So the distribution of path through the group is same. With the running of the Markov chain, we will have more and more samples of $(w(x_1), w(x_2), \dots, w(x_n))$, which can be used in the estimation of $E_{T_{n-1}} \frac{1}{w_1(y) + T_{n-1}}$. Note that $w(y)$ may not appear in the samples, so T_{n-1} can be calculated approximately by

$$\hat{T}_{n-1} = \frac{n-1}{n} \sum_{j=1}^n w(x_j)$$

Suppose now we have obtained m' samples of $(w(x_1), w(x_2), \dots, w(x_n))$, so we have m' \hat{T}_{n-1} : $\hat{T}_{n-1}^1, \hat{T}_{n-1}^2, \dots, \hat{T}_{n-1}^{m'}$. Then

$E_{T_{n-1}} \frac{1}{w_1(y) + T_{n-1}}$ can be estimated by

$$\frac{1}{m'} \sum_{j=1}^{m'} \frac{1}{w_1(y) + \hat{T}_{n-1}^j}$$

So now the probability $p^*(y)$ can be completely calculated, and this gives just the proposal function for the extrapolation step.

Similarly, the proposal function for projection can be obtained by reconstructing a path through all points of neighbours of point k except k .

5.3 Solving TSP Using the Weighted Markov Chain Method

In the beginning of this chapter, we have commented that the state space of the Markov chain can be augmented to be (k, L_k) , where k represents the number of cities of the state and L_k denotes the length of the tour at this state. In weighted Markov chain method, the state space is further augmented by a weighting factor w_k , so the state space is (k, L_k, w_k) instead of (k, L_k) . The chain is driven by weights. The distribution of (k, L_k) can be obtained after integrating out the weights w . In this way we can get the target distribution. Suppose there are N cities in the traveling salesman problem. So the problem can be divided into M levels, where M depends on the number of cities in the initial group, i.e. the dimensions of the lowest level. Let the number of cities in the initial group be I , so $M = N - I$. The simulation process can be summarized as follows.

0) Given the adding order of N cities, sample a tour L_I through the first ordered I cities. Choose initial weight w_I . (Essentially w_I can be chosen arbitrarily.) The current state is (I, L_I, w_I) . Set $i = I$.

1) Set $j = i \pm 1$ according to probability $q_{i,j}$, where $q_{I,I+1} = q_{N,N-1} = 1$ and $q_{i,i+1} = q_{i,i-1} = \frac{1}{2}$, if $I < i < N$.

2) If $j = i + 1$, we try to add the $(i + 1)$ th ordered city into the current tour.

If $j = i - 1$, we try to delete the i th city from the current tour.

3) Calculate the ratio

$$r = w_i \frac{f(L_j)T((j, L_j) \rightarrow (i, L_i)) q_{j,i}}{f(L_i)T((i, L_i) \rightarrow (j, L_j)) q_{i,j}} \quad (5.2)$$

$$= w_i \frac{\alpha_j \exp(-L_j/t_j) \exp(-l_i/t_i) \hat{E}_{l_i} q_{ji}}{\alpha_i \exp(-L_i/t_i) \exp(-l_j/t_j) \hat{E}_{l_j} q_{ij}}$$

$$= w_i \frac{\alpha_j}{\alpha_i} \exp(-(L_j - l_j)(\frac{1}{t_j} - \frac{1}{t_i})) \frac{\hat{E}_{l_i} q_{ji}}{\hat{E}_{l_j} q_{ij}}. \quad (5.3)$$

According to the weighted Markov chain rule, the proposed transition is accepted or rejected.

4) Go to step 1 until enough configurations have been obtained.

In step 3) α_i and α_j are the normalized constant of distributions $f(L_i)$ and $f(L_j)$, respectively.

$$f(L_i) = \alpha_i \exp(-L_i/t_i),$$

$$f(L_j) = \alpha_j \exp(-L_j/t_j).$$

l_i and l_j are the lengths of the sampled path through the neighbour group of point i and point j , respectively. \hat{E}_{l_i} , \hat{E}_{l_j} are the estimators of $E_{T_{n-1}} \frac{1}{w_1(y) + T_{n-1}}$ at state (i, L_i, w_i) and state (j, L_j, w_j) , respectively. t_i , t_j are the true temperatures of the two states, respectively.

In deriving the equalities above, the following relationship is used

$$L_j - l_j = L_i - l_i.$$

Considering the following figure 5, it is trivial because $L_j - l_j$ and $L_i - l_i$ are equal to the length of the part outside of the circle of the tour.

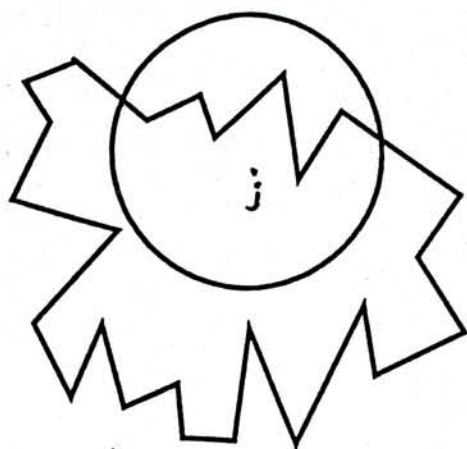


figure 5

The demonstrated figure of the tour not through point j

5.4 Temperature Scheme

In the simulation process, there are two kinds of temperatures: trial temperatures and true temperatures. The initial stages (in the buildup order) usually employ higher trial temperatures. The temperature is decreasing along the buildup order and stops at the lowest one (corresponding to the end stage of the buildup order). We have mentioned that a trial temperature is usually higher than the corresponding true temperature.

How to choose these temperatures?

In equation (5.3), the trial temperature is absorbed into the term $\frac{\hat{E}_{l_i}}{\bar{E}_{l_j}}$. For the true temperature, besides the part absorbed into $\frac{\hat{E}_{l_i}}{\bar{E}_{l_j}}$, it enters into the ratio r in the form:

$$\frac{1}{t_j} - \frac{1}{t_i} \quad j = i \pm 1$$

If $t_i = t_j$, the term $\exp(-(L_j - l_j)(\frac{1}{t_j} - \frac{1}{t_i}))$ will be canceled out. If $\frac{1}{t_j} - \frac{1}{t_i} = c$, c is a constant, such as 0.02, 0.01, etc. These temperature scheme will partially cancel out the influence of temperature jumping, and make the transition easier from one state to its adjacent state ($j = i \pm 1$).

5.5 How to Adjust the Constant Prior-ratio

It is a difficult problem to adjust the ratio $\frac{\alpha_j}{\alpha_i}$ correctly. The simplest method to determine the ratio is *trial and error*. The simulation starts at the higher temperature end and proceeds to the other one. Suppose the ratios $\frac{\alpha_2}{\alpha_1}, \frac{\alpha_3}{\alpha_2}, \dots, \frac{\alpha_k}{\alpha_{k-1}}$ have been determined so that the sampler for the corresponding distributions mix well and have a roughly even occupation numbers. We now want to determine a good ratio $\frac{\alpha_{k+1}}{\alpha_k}$ so that a sampler mixes well when $(k+1)$ th state is added to the Markov chain. Because of the order given above of adding the cities into tour, the changes of length L is smaller and smaller with the adding, and the ratio $\frac{\alpha_j}{\alpha_i}$ is close to 1 at the later states.

Because of the self-adjusting ability of weight w , the choice of ratios is not very critical for the transition from one state to its adjacent state in the running of the weighted Markov chain.

See Geyer and Thompson (1995) for a detail and similar discussion about how to determine their pseudo-priors, which is very critical for the simulated tempering process.

5.6 Validation of Our Algorithm by a Simple Example

100 random points were generated from $[0, 100]^2$ according to a continuous uniform distribution, and the first 10 ordered cities were chosen to test our algorithm. The numbers of points in the initial group and the neighbour group are both 5. The number of times the end state(10th ordered point) is visited is 5000. The results is listed in Table I.

Table I

true temperature	trial temperature	factorial ^a mean	average (resample n=20)
2.0	10.0	387.195	387.278(2.3875,61) ^b
	10.0 ~ 5.0 ^c		387.528(2.8174, 53)
5.0	10.0	392.97	392.087(8.0237,56)
	10.0 ~ 5.0		392.688(7.9234,81)
8.0	10.0	402.15	399.92(14.221,101)
	8.0		400.123(14.235, 99)

a: factorial mean is the weighted mean of all possible tour and $w_i \propto \exp(-l_i/t)$ $i = 1, \dots, 9!/2$

b: average(standard-deviation, independent sample size)

c: smoothly descend, $\frac{1}{t_{i+1}} - \frac{1}{t_i} = \frac{1}{60}$

We can see that there are no significant difference between the sample averages and the factorial means at every true temperature with different trial temperatures. But with increasing true temperature, the bias is increasing. With the increase of true temperature, the procedure assigns larger and larger weights to the longer tour which is difficult to sample using our method, because the sampled tour is obtained by adding one by one according to the given order instead of directly sampling from the highest dimensional space. In other words, certain unfavorable states are forbidden and then the sample averaging values is expected to be smaller than factorial weighted mean value. But this is favorable for the optimization problem.

5.7 Adding/Deleting by Blocks

Consider a random walk on the integers $1, 2, \dots, m$ having transitions to adjacent states with probability $\frac{p}{2}$ and staying at the same state with probability $1 - p$ for the interior points and $1 - \frac{p}{2}$ for the endpoints. In the terminology of Feller(1968) this is a random walk with reflecting barriers at $x = \frac{1}{2}$ and $x = m + \frac{1}{2}$. Using the methods of Feller(1968)

we find that the expected time to go from $x = 1$ to $x = m$ is $m(m - 1)/p$. We can conclude that using too many stages is inefficient. To reduce the number of stages, we can add/delete several cities (a *block*) at every stages.

Now the proposal moving is $(i, L_i, w_i) \rightarrow (i', L_{i'}, w_{i'})$, $i' = i \pm \text{block}$, and the proposal function is

$$T((i, L_i, w_i) \rightarrow (i', L_{i'}, w_{i'})) = \begin{cases} \prod_{j=0}^{\text{block}-1} T((i + j, L_{i+j}) \rightarrow (i + j + 1, L_{i+j+1})) & \text{adding} \\ \prod_{j=0}^{\text{block}-1} T((i - j, L_{i-j}) \rightarrow (i - j - 1, L_{i-j-1})) & \text{deleting} \end{cases} \quad (5.4)$$

Note the function $T(\cdot, \cdot)$ is independent of the weight w of states.

The corresponding ratio can be calculated by replacing proposal function of equation (5.2) by equation (5.4).

5.8 The Sequential Optimal Method and Post Optimization

For demonstrating the advantage of the stochastic optimal method, we first introduce another method for solving TSP— sequential optimal method. It is a deterministic method. At every step one city is added into the current tour using the Branch-and-Bound method to find a optimal path through the neighbour group of the city to be added, instead of sampling and resampling. Because the city is added deterministically, the added city need not be deleted from the current tour. The numbers of initial group points and the neighbour group size often affect the quality of solution. In our experience, for 100 to 500 points problem, usually a satisfactory solution can be found from pairs (m, m) , $11 \leq m \leq 16$. Of course, better solution can be found if m is enlarged, but this will take much more CPU time, because the time needed to find an optimal path through a neighbour group increases exponentially with the size of neighbour group even using the Branch and Bound method.

A 100-point problem, generated from Euclidean space $[0, 100]^2$ according to a continuous uniform distribution. Applying the *sequential optimal* method to this problem, a tour with length 777.9 is obtained (initial group size and neighbour group size are 12. Search from $10 \leq m \leq 16$).

Now applying *stochastic optimal* method to this problem again, the value of block is 5, within the first 200 reaching-end tours (Markov chain state is (N, L_N, w)), a tour with length 772.95 is found.

Because the city is added using sampling method one by one, the tour may not be optimal in some local path, although the framework of the whole tour is very good. *Post-optimization* procedure can be used to locally optimize the tour. The tour length is 768.8 after post optimization (the corresponding data and optimal tour figure can be found in Appendix B).

The procedure *post optimization* is as following,

1) Obtain the initial tour using stochastic method.

2) repeat for all cities in the order given by initial tour

2.1) delete the city from the current tour by just connecting its two adjacent cities , get a new tour with only $N - 1$ city.

2.2) add the city into the new tour by finding an optimal path through its neighbour group using Branch-and-Bound method .

Usually a better tour will be obtained after post optimizing the initial tour obtained using stochastic method.

5.9 Composite Algorithm

Using the algorithm described above, we can get the optimal or near optimal tour. Because of the order given in section (5.1), the change of the length is smaller and smaller with the adding of cities. At the same time the trial and true temperatures are becoming lower, so the cities are added into the current tour almost deterministicly. Considering the fact that the time for the Markov chain moving l steps away along the sequence of stages is proportional to l^2 , it is inefficient to simulate such a long Markov chain if we are only interested in the optimal solution of the problem. In our experience, you can obtain satisfactory solutions using the *composite algorithm*:

Now applying *stochastic optimal* method to this problem again, the value of block is 5, within the first 200 reaching-end tours (Markov chain state is (N, L_N, w)), a tour with length 772.95 is found.

Because the city is added using sampling method one by one, the tour may not be optimal in some local path, although the framework of the whole tour is very good. *Post-optimization* procedure can be used to locally optimize the tour. The tour length is 768.8 after post optimization (the corresponding data and optimal tour figure can be found in Appendix C).

The procedure *post optimization* is as following,

- 1) Obtain the initial tour using stochastic method.
- 2) repeat for all cities in the order given by initial tour
 - 2.1) delete the city from the current tour by just connecting its two adjacent cities , get a new tour with only $N - 1$ city.
 - 2.2) add the city into the new tour by finding an optimal path through its neighbour group using Branch-and-Bound method .

Usually a better tour will be obtained after post optimizing the initial tour obtained using stochastic method.

5.9 Composite Algorithm

Using the algorithm described above, we can get the optimal or near optimal tour. Because of the order given in section (5.1), the change of the length is smaller and smaller with the adding of cities. At the same time the trial and true temperatures are becoming lower, so the cities are added into the current tour almost deterministicly. Considering the fact that the time for the Markov chain moving l steps away along the sequence of stages is proportional to l^2 , it is inefficient to simulate such a long Markov chain if we are only interested in the optimal solution of the problem. In our experience, you can obtain satisfactory solutions using the *composite algorithm*:

- 1) About the first 20 – 40 percents ordered cities are added using stochastic method .
- 2) The remainder are added using sequential optimal method.
- 3) Use post-optimization method to optimize the last tour.

Usually in step 3 the improvement of the tour is not very large because more than about 60 percents of cities are added using sequential optimal method. One can sometimes omit it.

5.10 Numerical Comparisons and Tests

To assess the efficiency of the sequential optimal and the stochastic method, and to make comparisons with existing methods , we have carried out several series of computational tests both on randomly generated problems and on TSPs described in the operation research literature. The first set of problems were derived by generating n points in $[0, 100]^2$, according to a continuous uniform distribution, and by using the symmetric Euclidean distances between these points. The various procedures used in the computational tests on the same data set are:

1) Sequential optimal: initial group size and neighbour group size are equal and vary in the range $10 \leq p \leq 15$.

2) Composite method: the first 40 ordered cities are added using stochastic method, the other 60 cities are added using sequential optimal method. For every starting configuration of 10 cities we sample, the program ends when it has obtained 5 final tours. Run the procedure 10 times independently, and then select the best one from the total 50 tours.

The parameters setting of the procedure is as follows: initial group size is 10, neighbour group size is 10, block is 5, resample from 100 independent samples, temperature scheme: begins from 5.0 then decrease by the equation $\frac{1}{t_{i+1}} - \frac{1}{t_i} = \frac{1}{100}$. The average total time is 100 seconds on an Indy IP22 workstation.

3) Simulated Annealing: run SA(programme can be found in Press et al 1987) 9 times independently, and select the best one from the 9 final tours. Temperature begins from 0.5

and then decreases 150 steps by factor 0.95. maximum number of successful path changes before continuing is 1500, and the maximum number of paths tried at every temperature is 15000. The average total time is about 100 seconds On an Indy IP22 workstation.

Table II

Average solution costs for random Euclidean problem over 100 trials

procedure	p	n=100	n=500
GENIUS	5	791.2	1704.3
	6	788.3	1701.0
	7	788.6	1698.6
SA		787.4	
Sequential	$10 \leq p \leq 15$	778.7	
	$11 \leq p \leq 16$		1691.5
Stochastic		777.0	

In additional to these tests on randomly generated problems, we have compared our methods with other algorithms on the following problems, where alternative heuristics had already been tested and for which optimal values were available:

K24: a 100-vertex problem(number 24) described in Krolak, Felts and Marble(1971).

Grid100: a 100-vertex grid problem generated as in Cerny(1985).

Grid400: a 400-vertex grid problem generated in the same fashion.

Grid1600: a 1600-vertex grid problem generated in the same fashion.

G442: the Grötschel et al(1989) 442-vertex problem.

P532: the Padberg and Rinaldi(1987) 532-vertex problem.

All problem are solved with composite method (about 40 percents of points are added stochastically, and the other are added using sequential optimal method, and then optimize the tour using post-optimization method). These results are reported in Table

III.

Table III

Computational Results for Problems Described in the Operations Research Literature

Procedure	K24	Grid100	Grid400	Grid1600	G442	p532
2-Opt ^a		103.3			54.67	
Lin-Kernighan ^a					52.54	
CCAO ^b	21320	102.5	419.1		52.79	28726
GENIUS ^b	21282	100 ^c	400 ^c		51.27	28175
SA		100 ^d			51.42 ^e	28418
Tabu Search ^f					51.10	27839
Genetic					51.21 ^g	
Composite	21282 ^h	100 ^h	400 ^h	1600 ⁱ	50.82 ⁱ	27699 ⁱ
Exact optimal value	21282	100	400	1600	50.67	27686

a: Reinelt(1992);

b: Gendreau et al (1992);

c: the neighbour group size is 3, and restrict the path to be vertical or horizontal line;

d: Rossier, Troyon and Liebling(1986);

e: Troyon(1988);

f: Fiechter(1990);

g: Liu et al (1995);

h: the best solution of 100 trials;

i: the best solution of 500 trials.

It is evident that the composite method compares well with all existing methods. The corresponding optimal tours we found are plotted in Appendix C.

Chapter 6

Conclusion

We have described a new Markov chain construction method, weighted Markov chain Monte Carlo, for simulation based on auxiliary variable. We have successfully applied it to the TSP. Computational experiments carried out both on randomly generated problems and on problems described in the operations research literature confirm the relative efficiency of the method. Weighted Markov chain Monte Carlo method is a general method. It can be applied widely to simulation problems involving multimodal surface and to simulation and optimization problems with buildup structures.

REFERENCES

- Aarts, E. and Korst, J. (1989) Simulated Annealing and Boltzman Machines: a stochastic approach to combinatorial optimization and neural computing, Wiley, Chichester.
- Besag, J. and Green, P.J. (1993) Spatial statistics and Bayesian computation, *J. R. Statist. Soc. B*, 55, 25-37.
- Bonomi, E., and Lutton, J.L. (1984) The N -city traveling salesman problem: Statistical mechanics and the Metropolis algorithm, *SIAM review* 26, 551-568.
- Brady, R.M. (1985) Optimization strategies gleaned from biological evolution, *Nature* 317, 804-806.
- Carpaneto, G., and Toth, P. (1980) Some new branching and bounding criteria for the asymmetric traveling salesman problem, *Management Science* 26, 736-743.
- Cerny, V. (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, *J. Optim. Theory Appl.* 45, 41-51.
- Clarke, G., and Wright, J.W. (1963) Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research* 12, 568-581.
- Cuticchia, A.J., Arnold, J. and Timberlake, W.E. (1992) The use of simulated annealing in chromosome reconstruction experiments based on binary scoring, *Genetics* 132, 591-601.
- Dantzig, G.B., Fulkerson, D.R., and Johnson, S.M. (1954) Solution of a large scale traveling salesman problem, *operational Research* 2, 393-410.
- Feller, W. (1968) An introduction to probability theory and its application (Vol. 1, 3rd ed., rev.), John Wiley, New York.
- Fiechter, C.N. (1990) A parallel tabu search algorithm for large scale traveling salesman problem, working paper 90/1, Département de Mathématiques, École Polytechnique Fédérale de Lausanne.

REFERENCES

- Flood, M.M. (1956) The traveling salesman problem, *Operations Research* 4, 61-75.
- Gelfand, A.E., and Smith, A.F.M. (1990) Sampling -based approaches to calculating marginal densities, *J. Amer. Statist. Ass.* 85, 398-409.
- Geman, S., and Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, 721-741.
- Gendreau, M., Hertz, A., and Laporte, G. (1992) New insertion and post-optimization procedures for the traveling salesman problem, *Operations Research* 40, 1086-1094.
- Geweke, J. (1989) Bayesian inference in econometric models using Monte Carlo integration, *Econometrica* 57, 1317-1339.
- Geyer, C.J. and Thompson, E.A. (1995), Annealing Markov chain Monte Carlo with application to ancestral inference. *J. Amer. Statist. Assoc.* To appear.
- Gilmore, P.C., and Gomory, R.E. (1964) Sequencing a one state-variable machine: A solvable case of the traveling salesman problem, *Operations Research* 12, 655-679.
- Gilmore, P.C., Lawler, E.L.M and Shmoys, D.B. (1985) Well-solved special cases, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.), *The traveling salesman problem. A guided tour of combinatorial optimization*, Wiley, Chichester, 87-143.
- Glover, F. (1989) Tabu search, part I, *ORSA Journal on computing* 1, 190-209.
- Glover, F. (1990) Tabu search, part II, *ORSA Journal on computing* 2, 4-32.
- Glover, F. (1993) A user's guide to tabu search, *Annals of Operations Research* 41, 3-28.
- Glover, F., and McMillan, C. (1986) The general employee scheduling problem: An integration of MS and AI, *Computers & Operations Research* 13, 563-573.
- Golden, B.L., and Skiscim, C.C. (1986) Using simulated annealing to solve routing and location problems, *Naval Research Logistics Quarterly* 33, 261-280.

- Golden, B.L., and Stewart, Jr., W.R. (1985) Empirical analysis of heuristics, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (eds.) *The Traveling Salesman Problem. A Guided Tour of Combinational Optimization*, Wiley, Chichester, 207-249.
- Grötschel, M., Jünger, M. and Reinelt, G. (1989) Via minimization with pin preassignments and layer preference, *Z. Angew. math. mech.* 69 393-399.
- Hastings, W.K. (1970) Monte carlo sampling methods using Markov chain and their applications, *Biometrika* 57, 97-109.
- Holland, J. (1975) *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, MI.
- Irwin, M., Cox, N., and Kong, A. (1994) Sequential imputation for multilocus linkage analysis, *Proc. Nat. Acad. Sci. U.S.A.* 91, 11,684-11,688.
- Johnson, D.S. (1990) Local optimization and the traveling salesman problem, in: M.S. Paterson (ed.) *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, Lecture Note in Computer Science, Springer Verlag, Berlin, 446-461.
- Johnson, D.S., Papadimitriou, C.H., and Yannakakis, M. (1985) How easy is local search? *Proc. Annual Symposium on Foundations of Computer Science*, Los Angeles, 39-42.
- Kirkpatrick, S., Gelatt, Jr., C.D., and Vecchi, M.P. (1983) Optimization by simulated annealing, *Science* 220, 671-680.
- Kong, A., Liu, J.S., and Wong W.H. (1994) Sequential imputation and Bayesian missing data problems, *J. Amer. Statist. Assoc.* 89, 278-288.
- Krolak, P.D., Felts, W., and Marble, G. (1971) A man machine approach toward solving the traveling salesman problem, *Commun. ACM* 14, 327-334.
- Langevin, A., Soumis, F., and Desrosiers, J. (1990) Classification of traveling salesman problem formulations, *Operations Research Letters* 9, 127-132.

REFERENCES

- Laporte, G. (1992) The traveling salesman problem: an overview of exact and approximate algorithms, *European Journal of Operational Research* 59, 231-247.
- Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (1985) The Traveling Salesman Problem. A Guided Tour of Combinational Optimization, Wiley, Chichester.
- Lenstra, J.K., Rinnooy Kan, A.H.G. (1975), Some simple applications of the traveling salesman problem, *Operational Research Quarterly* 26, 717-733.
- Lin, S. (1965), Computer solutions of the traveling salesman problem *Bell System Computer Journal* 44, 2245-2269.
- Lin, S., and Kernighan, B.W. (1973), An efficient heuristic algorithm for the traveling salesman problem, *Operations Research* 21, 498-516.
- Liu, Y., Kang, L.S., and Chen, Y.P. (1995) Non-numerical parallel algorithms (Vol. 2): Genetic Algorithm, Academic Press (in Chinese).
- Marinari, E., and Parisi, G. (1992) Simulated tempering: A new Monte Carlo scheme, *Europhysics Letters* 19, 451-458.
- Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953), Equation of state calculations by fast computing machines, *Journal of Chemical Physics* 21, 1087-1091.
- Miller, D.L., and Pekny, J.F. (1991), Exact solution of large asymmetric traveling salesman problems, *Science* 251, 754-761.
- Nahar, S., Sahni, S., and Shragowitz, E. (1989) Simulated annealing and combinatorial optimization, *International Journal of Computer Aided VLSI design* 1, 1-23.
- Ong, H.L., and Huang, H.C. (1989), Asymmetric expected performance of some TSP heuristics, *European Journal of Operations Research* 43, 231-238.
- Padberg, M.W., and Rinaldi, G. (1987) Optimization of a 532-city symmetric traveling salesman problem by branch and cut, *Opns. Res. Letts.* 6, 1-7.

- Peskun, P.H. (1973) Optimum Monte Carlo sampling using Markov chains, *Biometrika*, 57, 97-109.
- Reinelt, G. (1992) Fast heuristics for large geometric traveling salesman problems, *ORSA J. Comput.* 4, 206-217.
- Ripley, B.D. (1987) *Stochastic Simulation*, Wiley, New York.
- Rosenkrantz, D.J., Stearns, R.E., and Lewis, II, P.M. (1977) An analysis of several heuristics for the traveling salesman problem, *SIAM Journal on Computing* 6, 563-581.
- Rossier, Y., Troyon, M., and Liebling, T.M. (1986) Probabilistic exchange algorithms and the Euclidian traveling salesman problem, *Opns. Res. Spektrum* 8, 151-164.
- Stander, J., and Silverman B.W., (1994) Temperature schedules for simulated annealing, *Statistics and Computing* 4, 21-32.
- Tanner, M.A. (1991) Tools for statistics inference, observed data and data augmentation (with discussion), *Lect. Notes Statist.* 67, Springer-Verlag, New York.
- Tanner, M.A., and Wong, W.H. (1987) The calculation of posterior distributions by data augmentation (with discussion), *J. Amer. Statist. Ass.* 82, 528-550.
- Tierney, L. (1994) Markov chains for exploring posterior distributions, *Ann. Statist.* 22, 1701-1762.
- Toda, M., Kubo, R., and Saitô, N. (1983) *Statistical Physics*, Springer-Verlag, Berlin.
- Troyon, M. (1988) Quelques heuristiques et Résultats Asymptotiques pour trois problèmes d'optimisation combinatoire, Thèse No. 754, École Polytechnique Fédérale de Lausanne, Switzerland.
- Wong, W.H., (1995) Comments for *Bayesian Computation and Stochastic Systems*, *Statistical Science* 10, 52-53.

Appendix A

THEOREM: If the following equality holds

$$\int g(x, w)w dw = cf(x). \quad (\text{A.1})$$

Let

$$h(x', w') = \int \int g(x, w)k((x, w) \rightarrow (x', w'))dw dx, \quad (\text{A.2})$$

then we have

$$\int h(x', w')w' dw' = cf(x'). \quad (\text{A.3})$$

proof. Suppose the current state of the weighted Markov chain is (x, w) , and the proposed transition state is (x', w') . If we accept the proposed transition, the next state is (x', w') . In this case the transition kernel function is

$$k((x, w) \rightarrow (x', w')) = \min\left(\frac{r(x, w; x')}{\theta}, 1\right)T(x \rightarrow x'), \quad (\text{A.4})$$

else if the proposed transition is rejected, the next state should be (x, w^*) . For convenience, in this case we replace x by x' and replace w by w' , so the transition kernel function is

$$k((x', w') \rightarrow (x', w^*)) = \int [1 - \min\left(\frac{r(x', w'; x'')}{\theta}, 1\right)T(x' \rightarrow x'')]dx''. \quad (\text{A.5})$$

Substitute (A.4) and (A.5) into the left side of (A.3), we get

$$\int h(x', w')w' dw' = I(x \rightarrow x') \int \int \int g(x, w)k((x, w) \rightarrow (x', w'))dx dw w' dw'$$

$$\begin{aligned}
& + I(x' \rightarrow x') \int \int g(x', w') k((x', w') \rightarrow (x', w^*)) dw' w^* dw^* \\
& = I + II
\end{aligned}$$

where $I(\cdot \rightarrow \cdot)$ is an indicate function.

$$\begin{aligned}
I & = I(x \rightarrow x') \left[\int \int_{\{r(x, w; x') \geq \theta\}} g(x, w) T(x \rightarrow x') w \frac{f(x') T(x' \rightarrow x)}{f(x) T(x \rightarrow x')} dw dx \right. \\
& \quad \left. + \int \int_{\{r(x, w; x') < \theta\}} g(x, w) T(x \rightarrow x') \frac{w f(x') T(x' \rightarrow x)}{\theta f(x) T(x \rightarrow x')} \theta dw dx \right] \\
& = I(x \rightarrow x') f(x') \int \int \frac{g(x, w) w}{f(x)} T(x' \rightarrow x) dw dx \\
& = I(x \rightarrow x') f(x') \int \frac{T(x' \rightarrow x)}{f(x)} dx \int g(x, w) w dw \\
& = cf(x') I(x \rightarrow x')
\end{aligned}$$

$$\begin{aligned}
II & = I(x' \rightarrow x') \int \int T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \\
& \quad - \int \int_{\{r(x', w'; x'') \geq \theta\}} T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \\
& \quad - \int \int_{\{r(x', w'; x'') < \theta\}} \frac{r(x', w'; x'')}{\theta} T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \\
& = I(x' \rightarrow x') \left[\int \int_{\{r(x', w'; x'') < \theta\}} T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \right. \\
& \quad \left. - \int \int_{\{r(x', w'; x'') < \theta\}} \frac{r(x', w'; x'')}{\theta} T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \right] \\
& = I(x' \rightarrow x') \int \int_{\{r(x', w'; x'') < \theta\}} \left(1 - \frac{r(x', w'; x'')}{\theta} \right) T(x' \rightarrow x'') g(x', w') w^* dx'' dw' \\
& = I(x' \rightarrow x') \int g(x', w') dw' \int_{\{r(x', w'; x'') < \theta\}} \left(1 - \frac{r(x', w'; x'')}{\theta} \right) T(x' \rightarrow x'') w^* dx'' \\
& = I(x' \rightarrow x') \int g(x', w') w' dw' \\
& = cf(x') I(x' \rightarrow x')
\end{aligned}$$

Where in (II), we let

$$q = \int_{\{r(x', w'; x'') < \theta\}} \left(1 - \frac{r(x', w'; x'')}{\theta} \right) T(x' \rightarrow x'') dx'' \quad (\text{A.6})$$

$$w^* = \frac{w'}{q} \quad (\text{A.7})$$

From (I) and (II), we have (A.3). So we have proved the theorem for the choice 1 in rejection step, i.e. return to state $(x, w/q)$.

For choice 2 in rejection step, we can prove the theorem as follows:

Suppose the first $(i - 1)$ th proposal transitions are all rejected, until the i th proposal transition is accepted. So then the distribution of i is a geometric distribution with parameter p . Let u_1 be a random number generated from uniform distribution $U(0, 1)$, so we have

$$\begin{aligned} p &= P(i = 1) \\ &= \int T(x \rightarrow x') P(r(x, w; x') \geq \theta \text{ or } u_1 < \frac{r(x, w; x')}{\theta}) dx' \\ &= \int T(x \rightarrow x') (1 - P(u_1 \geq \frac{r(x, w; x')}{\theta})) dx' \\ &= 1 - \int_{\{r(x, w; x') < \theta\}} T(x \rightarrow x') (1 - \frac{r(x, w; x')}{\theta}) dx' \\ &= 1 - q \end{aligned}$$

$$P(i = k) = q^{k-1} p$$

Suppose at last we accept the proposed state (x', w') , so that the corresponding transitional kernel is

$$\begin{aligned} k((x, w) \rightarrow (x', w')) &= \sum_{k=1}^{\infty} q^{k-1} [T(x \rightarrow x') I(r(x, w; x') \geq \theta) \\ &\quad + T(x \rightarrow x') \frac{r(x, w; x')}{\theta} I(r(x, w; x') < \theta)]. \end{aligned}$$

Then

$$\begin{aligned} \int h(x', w') w' dw' &= \int \int \int \sum_{k=1}^{\infty} q^{k-1} [T(x \rightarrow x') I(r(x, w; x') \geq \theta) \\ &\quad + T(x \rightarrow x') \frac{r(x, w; x')}{\theta} I(r(x, w; x') < \theta)] g(x, w) w' dx dw dw' \\ &= \int \int \int g(x, w) \frac{w'}{p} dx dw dw'. \end{aligned}$$

APPENDIX A.

If we let

$$w' = \begin{cases} r(x, w; x')p & \text{if } r(x, w; x') \geq \theta \\ \theta p & \text{if } r(x, w; x') < \theta, \end{cases} \quad (\text{A.8})$$

Similarly from (I) except the term $I(x \rightarrow x')$, we know (A.3) holds.

APPENDIX B

The following data is generated according to a continuous uniform distribution [0,100] by [0,100]

71.971801	75.182348	13.599048	68.028809	80.495621	11.560411
64.534440	74.932096	4.287851	60.405286	76.024659	10.461745
65.382855	70.522172	2.948088	47.230445	74.327830	18.933683
65.861995	63.664663	10.806604	30.872524	70.955535	24.802393
63.081759	57.338176	6.039613	26.300851	71.547594	35.013886
68.013550	52.342296	7.550279	26.670125	78.923917	27.201147
65.077670	52.305673	12.790307	25.296793	88.750877	27.246925
58.143864	46.333201	12.421033	21.680349	89.327677	33.436079
45.295572	46.513260	15.588855	22.156438	93.646046	37.006745
44.587542	51.564074	16.977447	17.725150	99.496445	41.285440
40.232551	54.463332	18.454543	16.302988	97.967467	52.922147
37.791070	57.039094	10.052797	9.128086	82.604450	54.176458
37.287515	48.094119	5.041658	2.206488	83.053072	54.884487
31.513413	41.752373	9.738456	2.584918	83.034761	55.278176
23.129978	42.173528	10.049745	2.398755	78.820154	58.769494
20.163579	48.472549	18.454543	5.545213	79.357280	73.244423
16.885891	59.593493	19.586779	10.638752	80.980865	72.371593
26.178777	68.611713	27.582629	7.690664	83.947264	72.463149
29.139073	78.499710	33.515427	10.675375	94.305246	72.759178
31.113620	73.070467	26.718955	19.852290	90.307321	79.274880
36.533708	73.171178	29.050569	30.124821	91.293069	88.027589
40.260018	69.179357	44.718772	32.618183	86.141545	93.704031
52.821436	70.622883	53.950621	21.057772	78.841517	89.043855
58.555864	82.641072	51.435896	20.889920	71.614734	79.625843
61.122471	95.629749	42.976165	19.483016		
49.153111	98.248238	40.720847	16.834010		
46.055483	97.662282	42.616047	13.409833		
46.812342	95.196387	42.597736	13.254189		
41.938536	91.659291	44.325083	10.220649		
44.138920	89.910581	50.373852	7.513657		
44.627216	82.332835	51.707511	12.210456		
37.980285	89.452803	59.492782	10.003967		
31.592761	94.787439	77.477340	1.016266		
31.675161	98.553423	80.001221	4.010132		
13.498337	97.918638	86.700034	1.391644		
11.673330	92.672506	87.130345	1.797540		
2.700888	91.576891	89.486373	0.857570		
14.374218	69.841609	92.892239	9.912412		

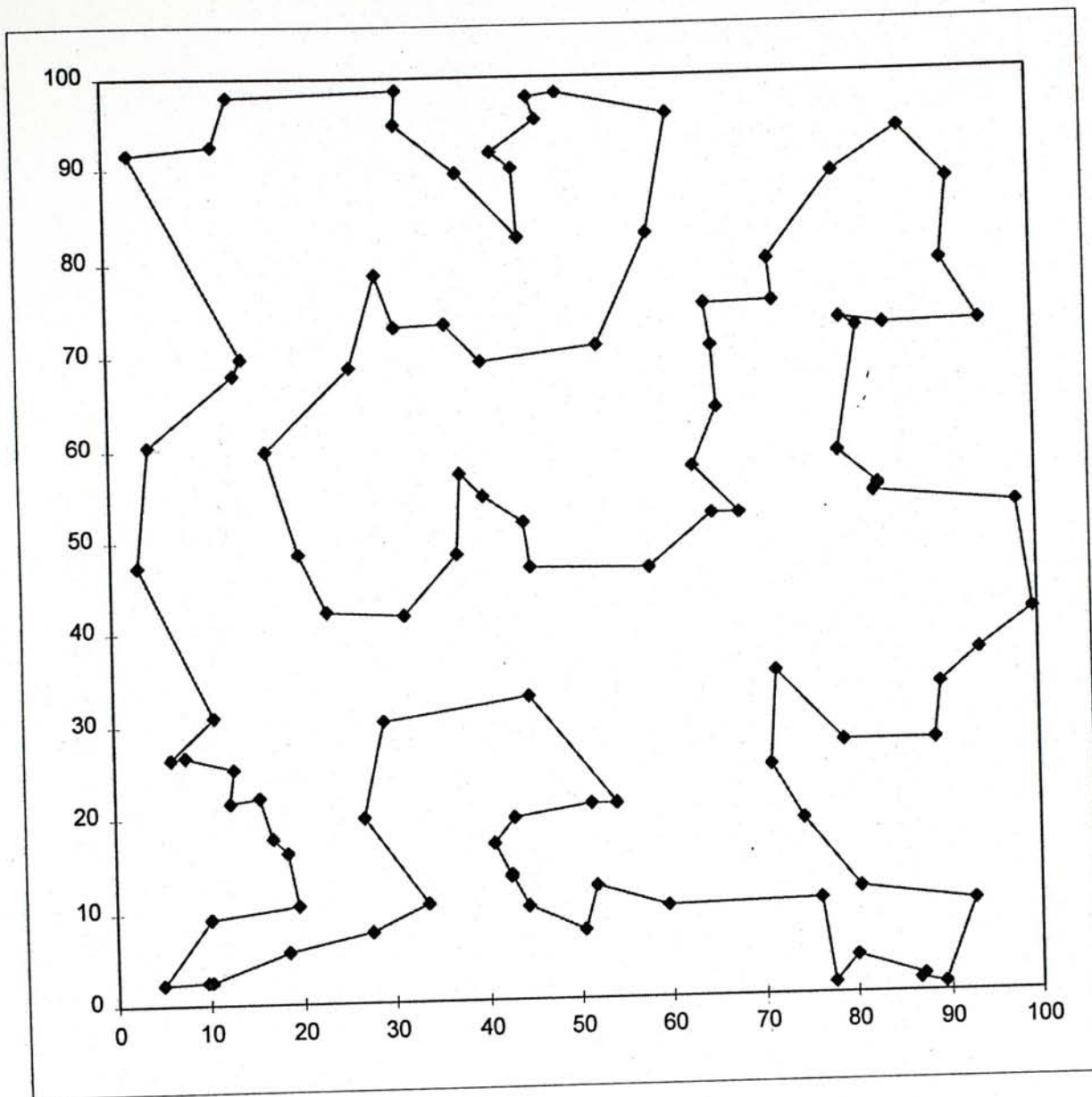


figure B.1

The near optimal tour of a 100-random-point-problem generated from uniform distribution $[0,100]$ by $[0,100]$ found by Stochastic algorithm, its tour length is 772.95

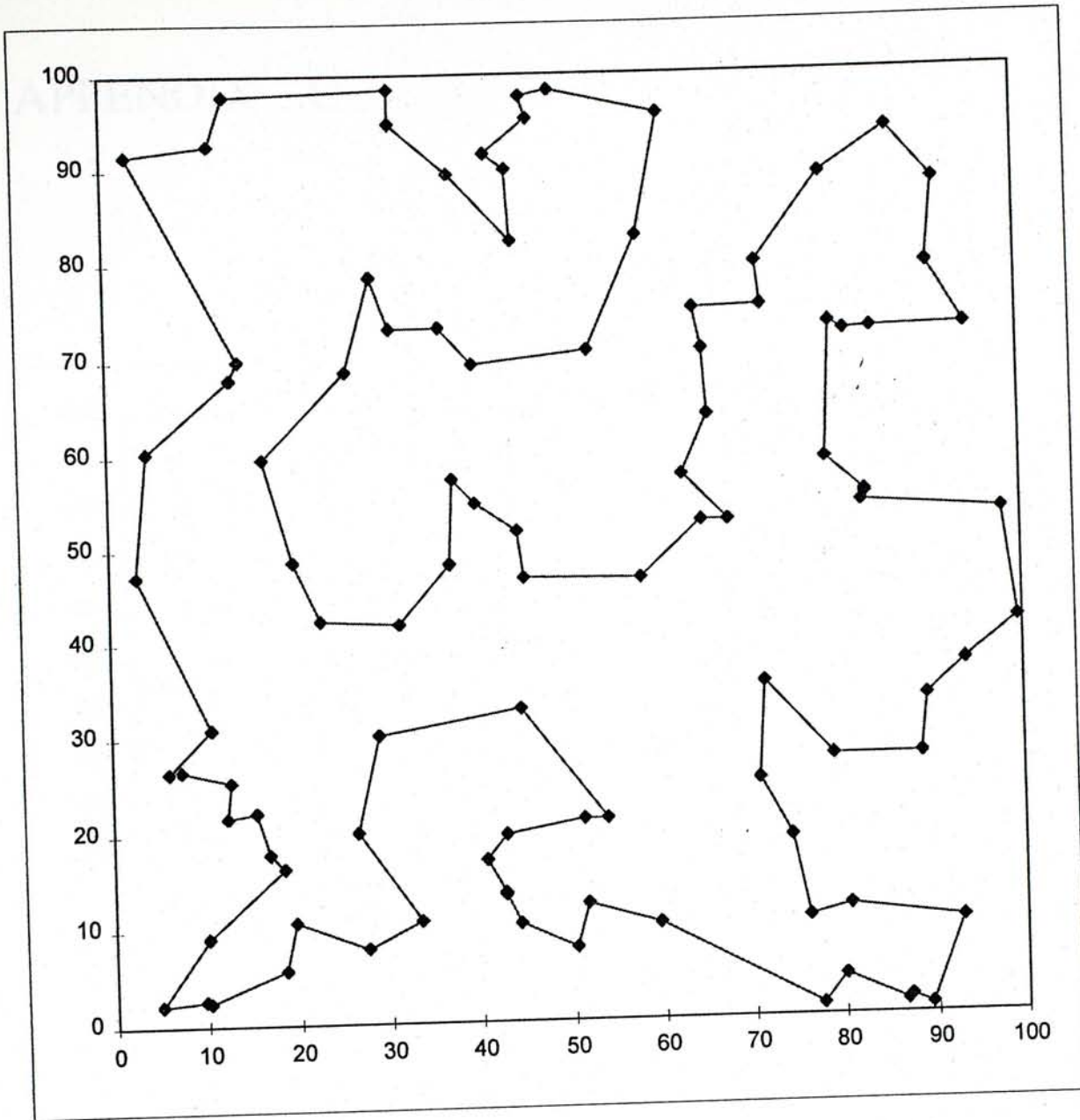


figure B.2

The near optimal tour of a 100-random-point-problem generated from Uniform distribution $[0,100]$ by $[0,100]$, its tour length is 768.8.

APPENDIX C

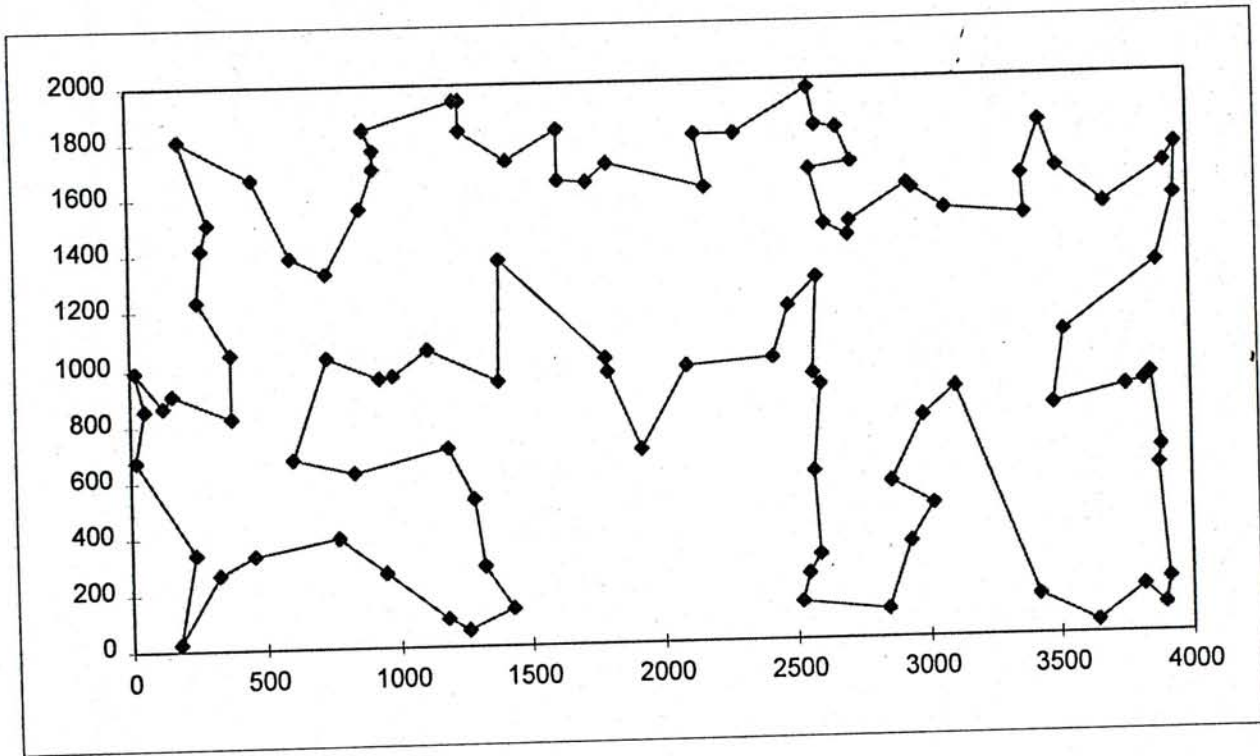


figure C.1
The exact optimal tour of k24 problem found by Composite algorithm
Its total length is 21282

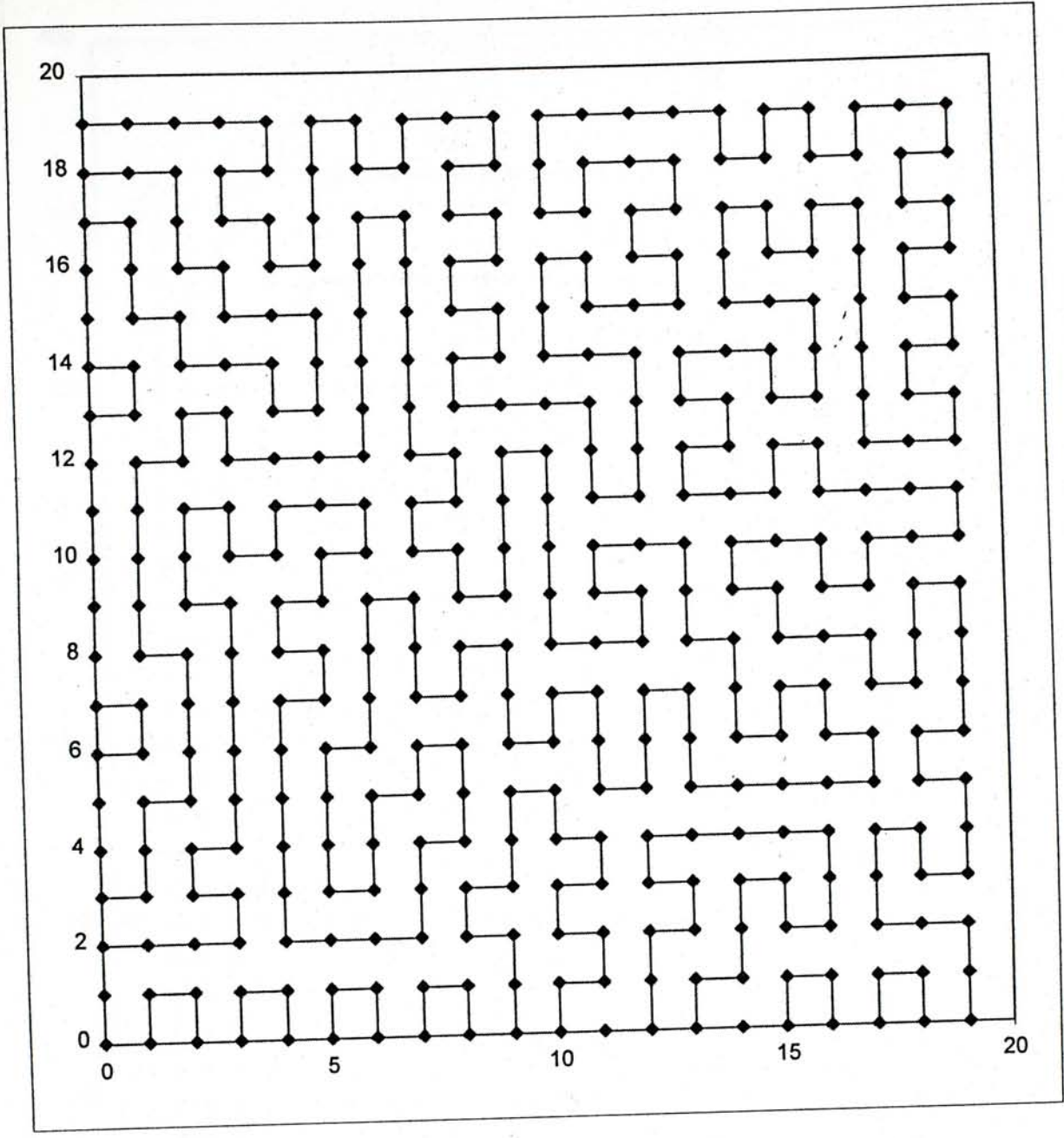


figure C.2

The exact optimal tour of Grid400 problem found by Composite algorithm

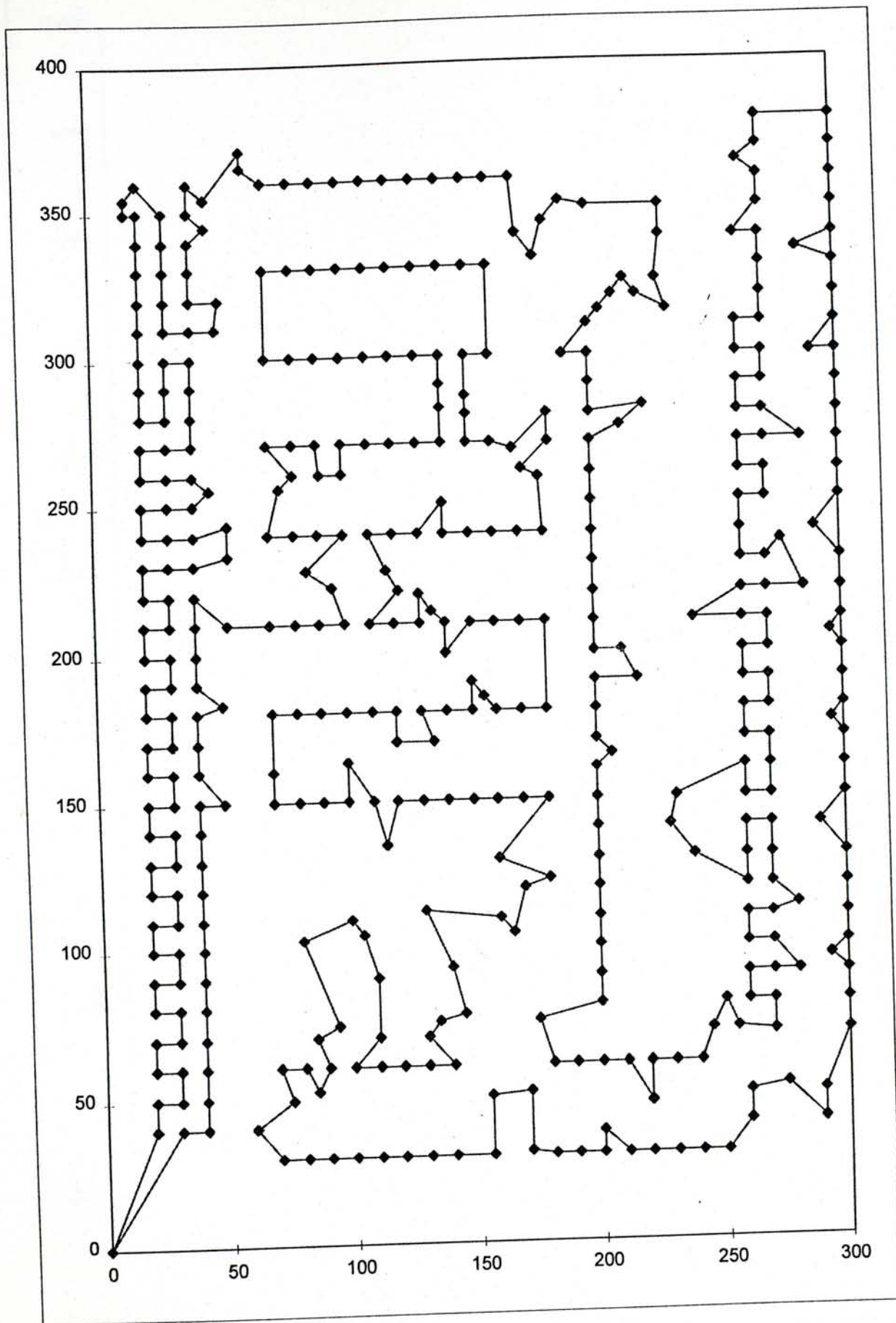


figure C.3

The near optimal tour of Grotchel-442 problem found by Composite algorithm, its length is 50.82.
 Exact optimal tour length is 50.76.

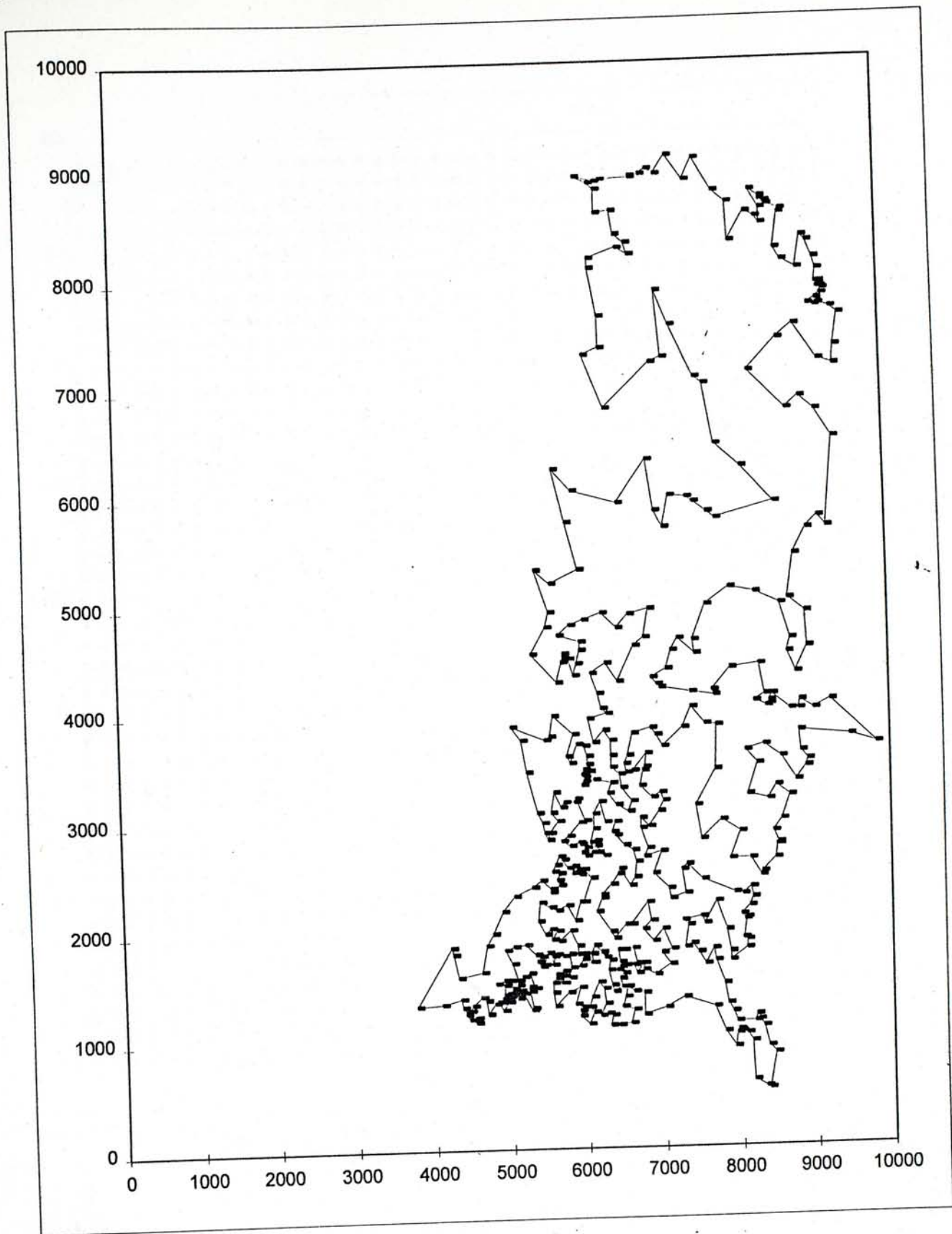


figure C.4

The near optimal tour of P-532 problem found by Composite algorithm.
Its total length is 27699, and the exact optimal value is 27686.

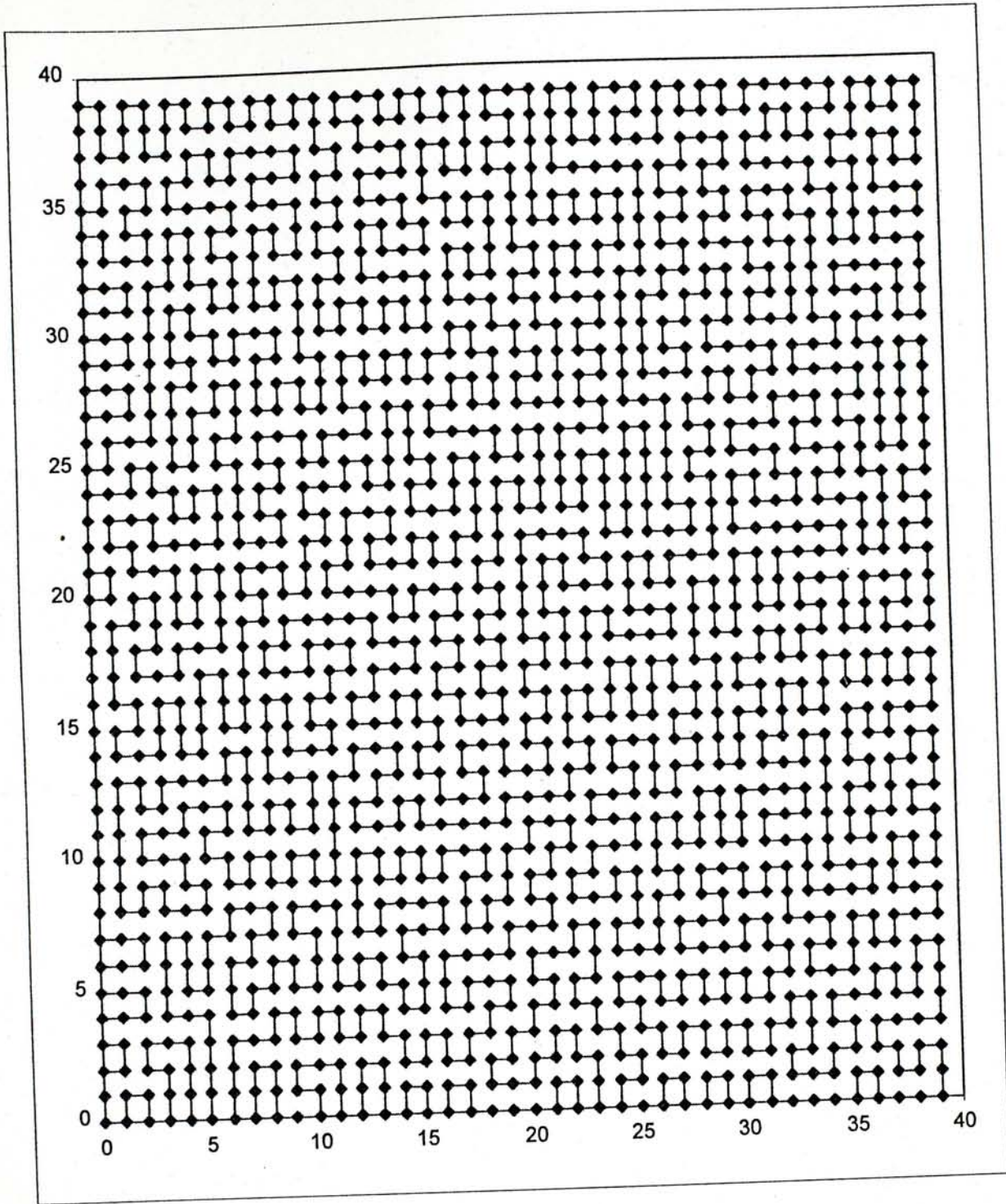
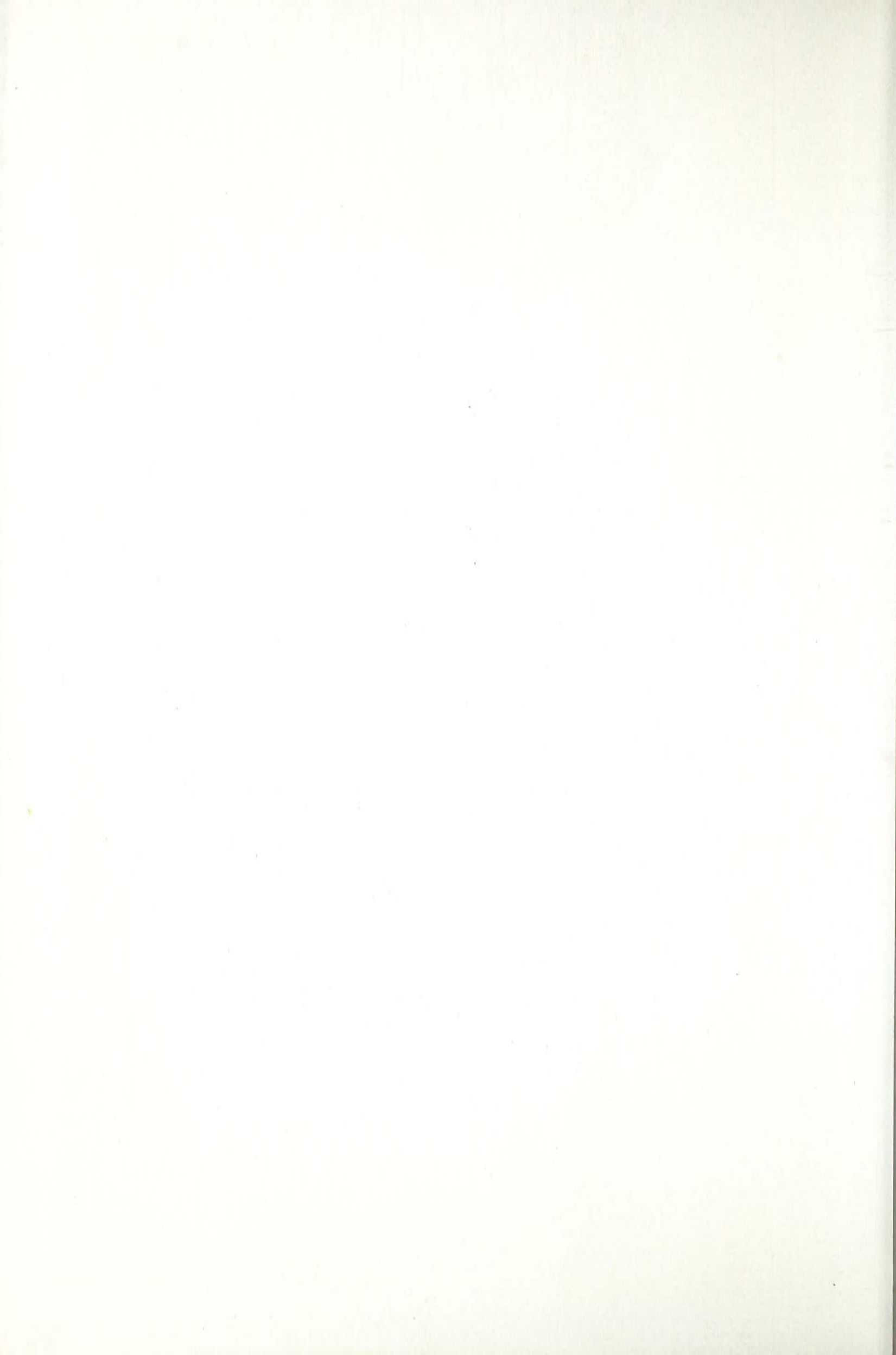


figure C.5
The exact optimal tour of Grid1600 problem found by Composite algorithm



CUHK Libraries



003510836