

1991

A heuristic using problem space perturbations for solving large traveling salesman problems

Andrew James Dvorocsik
Lehigh University

Follow this and additional works at: <https://preserve.lehigh.edu/etd>



Part of the [Industrial Engineering Commons](#)

Recommended Citation

Dvorocsik, Andrew James, "A heuristic using problem space perturbations for solving large traveling salesman problems" (1991). *Theses and Dissertations*. 5376.

<https://preserve.lehigh.edu/etd/5376>

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

**A Heuristic
Using Problem Space Perturbations
for Solving Large
Traveling Salesman Problems**

by

Andrew James Dvorocsik

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

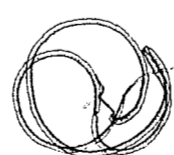
Master of Science

in

Industrial Engineering

Lehigh University

September 1990



Approved and recommended for acceptance as a
thesis in partial fulfillment for the degree of Master of
Science.

Sept 20 1990

(date)

Robert H. Storer

Professor R.H. Storer

Thesis Advisor

M.U. Thomas

Professor M.U. Thomas

Department Chairman

Table of Contents

Abstract	1
1. Introduction	2
1.1 Combinatorial Optimization	3
1.2 NP - Complete	5
1.3 Local Search	8
2. Traveling Salesman Heuristics	10
2.1 Tour Improvement Procedures	17
2.2 Probabilistic Local Search	19
3. Jiggling Algorithm	23
4. Experimentation	30
5. Summary	44
References	49
Vita	54

List of Tables

1. Space Filling Curve Comparisons	16
2. Comparison of 2-opt heuristics	37
3. Final Results	42
4. CPU Time Comparisons of Jiggling and 2-opts	43
5. Comparison of Jiggling to Standard Procedures.....	46

List of Figures

1. Tour Length vs Iterations	33
2. Comparison of 2-opt heuristics	38
3. Jiggling Algorithm Results	47
4. Comparison of Jiggling Results vs. 2-Opt	48

Abstract

One of the most heavily studied optimization problems in the field of operations research is the traveling salesman problem. Traditional local search approaches to solving this problem search in solution space, usually in a 2 or 3 interchanging neighborhood. In this thesis, a novel algorithm based on problem space search is examined.

This algorithm is based on the fact that tiny perturbations to the problem will, when operated on by a heuristic, generate a search neighborhood that yields better results than traditional neighborhoods. The "Jiggling" algorithm uses the space filling curve heuristic to generate the novel neighborhood that is defined by the perturbations.

The jiggling algorithm is found to be competitive with current methods. However, when jiggling and 2-opt are combined into a two stage heuristic, substantial improvement results.

1. Introduction

The Traveling Salesman Problem (TSP) is a classic optimization problem which although simple to describe, belongs to a category of problems which are very difficult to solve to optimality. This is true because the required solution time grows exponentially in the size of the problem. Problems such as this are said to belong to the class NP- Complete. The travelling salesman problem can be stated: given a group of cities and the distance between each city, find the shortest tour that visits each city once and only once and returns to the starting point. [Wasserman 1989]. This problem is trivial for a small number of cities, however, as the number of cities, n , increases the difficulty in solving the problem to optimality increases at the previously mentioned exponential rate. The traveling salesman is probably the most studied of the many combinatorial optimization problems. Much of this study has focused on heuristic solution techniques. The focus of this thesis is a TSP heuristic which operates by introducing perturbations to the problem, hence the term problem space search.

1.1 Combinatorial Optimization

Combinatorics is the study of arrangements: given a set of items, e.g. cities, which way should the cities be arranged so that the tour length is minimized. A subgroup of combinatorics is discrete or combinatorial optimization. Combinatorial optimization is the analysis of problems which are modeled as either the minimization or maximization of a value measure over a feasible space involving mutually exclusive, logical constraints. In the case of the traveling salesman problem, the objective measure is a minimization of the tour length, i.e. total distance travelled between cities, while the constraints are that each city must be visited once and only once and that the solution is a valid tour. Other combinatorial optimization problems include: The postman's problem, the knapsack problem, the maximum flow problem and various machine scheduling problems.

These problems, however, have not received the attention that the traveling salesman has. This is, no doubt, because of the simplicity of the problem as well as its many applications including basic routing

problems, VLSI chip design and printed circuit board design.

There are several types of traveling salesman problems. Their differences are based on the properties of the distances or costs between nodes. Given a cost matrix C , the distance between the cities can be represented by c_{ij} . A TSP is said to be symmetric if $c_{ij} = c_{ji}$, i.e. one whose distance from one city to another is always the same amount no matter what order the cities are visited. A non-symmetric TSP would be one whose costs are different depending on the order of visitation.

The next type of TSP is one in which the triangle inequality holds. This says that $c_{ik} \leq c_{ij} + c_{jk}$. This property is very important for many of the tour construction heuristics. A planar TSP is simply one whose node coordinates exist in a plane.

An important consideration for planar traveling salesman problems is the way in which the costs, or distances, are measured. One method is the taxi cab metric. In this method the distance between nodes is calculated as the sum of the absolute differences in the x and y coordinates. In the Euclidean metric, these distances are calculated using Euclidean distances, the

square root of the sum of the squares of the differences of the x and y coordinates.

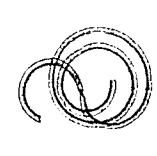
In this thesis a new heuristic for the planar TSP will be described. All of the problems will be created by filling a unit square with a specified number of randomly chosen cities. The Euclidean metric will be used for study, however, the proposed methods will work for any metric.

1.2 NP- Complete

Combinatorial Optimization problems are difficult to solve due to the tremendous size of the solution space, and the fact that each solution must be investigated (either implicitly or explicitly) to guarantee optimality. The size of the solution space grows exponentially with the problem size. The traveling salesman problem of size n cities contains $\frac{n!}{2n}$ unique tours. To evaluate all of these tours would literally take forever. When n is only 30 there exists $4.4 \cdot 10^{30}$ different tours. Considering the arithmetic required to compute a single tour, a 10-mflops computer would require $4.2 \cdot 10^{17}$ years. to evaluate all of the tours (Davis 1989). Clearly, one must use a different method

than trying every possible combination. Hence, the need to study combinatorial optimization. While much progress has been made in implicit enumeration approaches, e.g. Held and Karp (1970), the largest problems which have been solved optimally are limited to a few hundred cities. Traditionally, heuristics and algorithms have been used to 'solve' large traveling salesman problems rather than enumeration methods. A heuristic is a method of arriving at a solution using rules of thumb. An example would be to always travel next to the unvisited city closest to the current city, and repeat until all cities have been selected.

Optimization problems can be divided into three classes, the optimization version, the evaluation version and the recognition version. These versions can be defined as follows: Optimization version - given the problem find the optimal feasible solution; The Evaluation version - find the cost of the optimal solution; the Recognition version- given an instance of a problem is there a feasible solution? The recognition version requires only a yes or no answer. It can also be seen that the recognition version is no harder than the evaluation version, which itself is no harder than the optimization version.



One can now use these definitions to determine whether a problem falls into the category of P or NP. A problem is said to be in the class P if it can be solved by a polynomial-time algorithm. An example of such a problem would be the graph connectedness problem. Given a graph G , is G connected? This question can be easily answered in polynomial time using an efficient heuristic.

A problem which can not be solved using a polynomial time algorithm but for which there exists a polynomial time algorithm to verify a 'yes' answer to the problem is said to be in the class of NP problems. The traveling salesman problem has been shown to be in NP (Garey and Johnson 1979).

A recognition problem that is known to be in NP is said to be NP complete if all problems in NP polynomially transform to that problem. These two properties have been observed for the traveling salesman and so the TSP is in NP complete. This is important to recognize because there is no known algorithm which will optimally solve large traveling salesman problems in polynomial time. Knowing this, heuristics which try to obtain the best possible solution are the only reasonable alternative.

1.3 Local Search

As a reasonable alternative to complete enumeration a common heuristic known as local search is often used. A local search is simply a trial and error process which looks at a neighborhood of solutions in search of a better one. These methods generally run until no further improvement can be made and a solution is accepted to be at best optimal and at least locally optimal. A solution is locally optimal if the solution is the best in the surrounding neighborhood, where a neighborhood is loosely defined to be a solution set that contains solutions which are "close" to the current solution. The term local search comes from the idea of searching such a local neighborhood of solutions.

Within local search there are two primary search strategies. They are hillclimbing and steepest descent. In hillclimbing a new solution is compared to the current solution and accepted if the new solution is better. In steepest descent, all neighboring solutions are evaluated before the best is chosen as the next incumbent. These solutions are chosen from a

"neighborhood" of solutions which borders on the current solution. The term neighborhood refers to a collection of solutions which are "close" to a current solution.

Often, the most important part of a local search is the neighborhood definition. One can choose a large neighborhood at each step, but this requires a greater amount of searching. Alternatively, one can choose a small neighborhood which takes less time to search, but the chance of finding an improved solution decreases. This is because a search has a greater chance of getting "stuck" in a local optimum.

The key to local search is the specification of the neighborhood. In this thesis, a novel neighborhood for the planar TSP is investigated. This neighborhood is created by introducing small perturbations in the problem space. The conjecture is that the perturbations will provide a search space that is useful in finding good solutions.

2. Traveling Salesman Heuristics

Since an optimal solution can not be found for large problems, researchers have used many heuristics algorithms to find solutions for TSP problems. These heuristics fall into three categories. The first are tour construction procedures which generate a single solution using common sense rules. The second class are termed tour improvement. These are self-descriptive in that they attempt to find a better tour by improving upon an initial tour. The last group is composite procedures. These procedures are a hybrid of the first two classes where a better than random initial solution is used and then a tour improvement process is applied to obtain an even better solution. This composite procedure seems to work better than the other two for the obvious reason that an improvement over a good starting solution is better than merely a good starting solution or an improvement over a random solution.

A fundamental tour construction procedure is the nearest neighbor heuristic. This algorithm is intuitive.

1. Start with any city as incumbent.

2. Find the incumbent city's nearest neighbor, i.e. the closest city that is not already in the tour, and add that city to the tour as the incumbent.
3. Goto step 2 until all cities have been visited.
4. Connect the first and last cities.

Rosenkrantz, Stearns and Lewis (1974) researched this procedure and found that the worst case behavior for the nearest neighbor heuristic is $0.5 \cdot [\log(n)] + 0.5$.

Another common group of procedures for constructing tours are the insertion procedures. These include: nearest insertion, cheapest insertion, arbitrary insertion, and farthest insertion. An insertion procedure takes a subtour and chooses (inserts) a city which is not in the tour. This choice can be made on the basis of various criteria, hence, the various procedures.

Briefly, the nearest insertion procedure chooses the next unvisited city closest to the current subtour. This is repeated until all cities have been included in the tour. The next common insertion algorithm is the cheapest insertion procedure. This differs slightly from the nearest insertion in that the added area generated by two new arcs must be minimized. The farthest insertion procedure starts from a degenerate tour and

repeatedly chooses the non-tour city with the maximum distance to its nearest neighbor among the subtour cities. These insertion procedures do a relatively good job - roughly 25% over the Held- Karp lower bound (Johnson 1990).

The Held- Karp lower bound is a lower bound that is created by an iterative process which is based on the relationship between the symmetric traveling salesman problem and the minimum spanning tree problem. This approach is based on the idea that a minimum weight 1-tree is easy to compute. At each iteration the 1-tree problem is revised by adding and changing node weights such that the lower bound will increase.

Two algorithms worthy of separate note are the Double Minimum Spanning Tree and the Christofides Heuristic, which is a variant of the Double MST. The Double Minimum Spanning Tree Heuristic constructs a graph consisting of a minimum spanning tree for the cities. Then it constructs an Euler Cycle (a graph in which each edge is traversed exactly once (Minieka 1978)). A tour is then derived by traversing the cycle and taking the cities in the order in which they are first encountered. The Christofides' Heuristic is similar in that a minimal spanning tree must be constructed as

the first step. The second step is to identify all of the odd degree nodes in that spanning tree. It proceeds by solving a minimum cost perfect matching on the odd degree nodes using the original cost matrix. It then adds the branches from the matching solution to the branches already in the minimum spanning tree therefore obtaining an Euler cycle. The result is a subgraph with all nodes having an even degree. The final step is to remove polygons over the nodes with degree greater than 2 thus transforming the Euler cycle into a Hamiltonian Cycle. It has been found (Golden 1979) that the worst case behavior of the Christofides Heuristic is 1.5, and is generally in the area of 9% over the Held- Karp lower bound (Johnson 1990).

These algorithms and heuristics are used to find tours in all types of traveling salesman problems. The following algorithms can only be used when the cities can be represented as points on a plane. These algorithms include the Strip, Space Filling Curve, Recursive Clustering and Karp's Partitioning Algorithm. The strip algorithm divides the unit area into \sqrt{n} strips. A tour is then constructed by moving down the leftmost strip, ordering the cities by height. Once the first strip is completed move up the second strip and

then down the third, and so on, until all of the strips are completed. The final city is then connected to the first to complete the tour. This method usually gives a result roughly 30% over the Held Karp lower bound but performs much more poorly on real-world instances than on uniform random problems (Johnson 1990).

The space filling curve (SFC) heuristic creates a tour by visiting cities in the order in which they appear along a space filling curve. This method, first suggested by Bartholdi and Platzman (1985) is known to be the fastest tour construction heuristic (Johnson 1990). The SFC maps points in two dimensional Euclidean space into a single dimension. Any point on a unit square can be mapped to the unit interval via a space filling curve. The unit square is divided into four quadrants and each quadrant is then divided many times over to give many tiny regions. These regions are then ordered recursively to provide a transformation that orders each city. This ordering is then the tour. The tours that are created by the space filling curve heuristic are usually about 25% over the expected optimal. The expected optimal is found to be $.756*\sqrt{N}$, and the expected space filling curve distance is $.956*\sqrt{N}$ (Bartholdi and Platzman

1985). A comparison of expected and computer generated SFC values is contained in table 1. In this table, it can be seen that the average tour lengths generated for the test problems are consistent with Bartholdi and Platzman's expected values.

Because of its speed and good performance, the space filling curve heuristic is used to provide the initial solution before the jiggling procedure is implemented in the research. The space filling curve works because cities are grouped together and then arranged in a logical order. That is, cities which are close in the plane tend to be close after being mapped to the unit interval. A principle advantage of this heuristic is its speed. This is because once the location values for each node are determined, only one sort is required. A drawback to the SFC is that in the worst case solutions can be six times the optimal (Bartholdi and Platzman 1987).

Litke has suggested a Recursive Clustering algorithm which is a divide and conquer algorithm that gathers points which are close to each other into clusters and then applies an exhaustive search to find the optimal tour within a particular cluster (Litke 1984).

SFC COMPARISON

CITIES	EXP OPTM	EXP SFC	CALC SFC	CALC/EXP
1000	24.1914	30.2314	30.3008	1.0023
2000	34.2118	42.7536	43.3657	1.0143
3000	41.9008	52.3623	53.4592	1.0209
4000	48.3828	60.4627	60.8467	1.0064
5000	54.0937	67.5994	68.6908	1.0161
6000	59.2566	74.0514	74.2456	1.0026
7000	64.0045	79.9847	80.4646	1.0060
8000	68.4237	85.5072	87.6996	1.0256
9000	72.5743	90.6941	90.4870	0.9977
10000	76.5000	65.6000	95.9812	1.4631

This table shows the expected and calculated values of the Space Filling Cur

Table 1

The last of the standard construction algorithms is the Karp Partitioning Algorithm. This is a decomposition algorithm in which the cities are partitioned recursively by cuts through median cities until no more than a given number of cities are in any single partition. Dynamic programming is then applied to find the optimal subtour for each partition and the partitions are then connected to form a complete tour.

Of all of these unit square techniques, Litke's Clustering Algorithm gives the best results, although it does require more time. Here, one finds the first indication that it is necessary to find a good balance between the speed of an algorithm and the results that are given by the algorithm. For example, the farthest insertion method will almost always give a better result than the space filling curve, but the time required to do so is many, many times greater (Johnson 1990).

2.1 Tour Improvement Procedures

Tour construction methods can be augmented using tour improvement procedures. These algorithms are usually local optimization techniques. They look to find a neighboring tour which is better than the current

tour. The best known tour improvement heuristics are the 2-opt, 3-opt and the Lin-Kernighan. All of these heuristics are based on branch exchange which means that edges in the current tour are swapped or reversed if a better tour is found. These are actually neighborhood definitions rather than heuristics because various neighboring solutions are evaluated.

In the 2-opt procedure two cities, A and C are chosen randomly. The distances from A to its successor city B and from C to its successor city D are evaluated. If swapping the arcs, i.e. A to D and C to B, instead of A to B and C to D give a shorter tour, then accept the new tour. The same general concept applies to three opt where three arcs are chosen and the various combinations of new arcs are evaluated to try to identify a shorter tour.

The Lin-Kernighan procedure goes beyond what would be the next normal progression - 4-opt. A new tour is a neighbor of the current tour if it is shorter and can be obtained by breaking a 3-opt and then performing a greedy search.

The 2-opt and 3-opt procedures can be modified by limiting the neighborhood search space. This could be done, for example, by choosing the first city

randomly and the second by choosing a city that is within a certain number of cities on the current tour. An example would be a modified 2-opt with a modifying value of 10. Once the first city is chosen the second city is chosen to be within 10 cities from the first on the current tour.

An improvement on this modified 2-opt is a converging modified 2-opt where initially, the modifying size is 100 then 50, 25 20 ... and so on. This allows the neighborhood size to shrink as the tour length is decreased, therefore improving the efficiency of the neighborhood search as an optima is approached. This is because during the first 2-opt, a neighborhood of arcs within 100 of each other on the tour is examined. This size is gradually reduced which shrinks the search neighborhood and allows the search to investigate only small changes in the tour.

2.2 Probabilistic Local Search

Up until now, new tours have only been accepted if their tour length is shorter than the current best tour. However, it has been suggested that occasionally accepting a tour whose length is longer than the

current tour may allow the search to escape local minima. The most common technique to accomplish this is Simulated Annealing. The general philosophy of simulated annealing is to thoroughly investigate the space of possible solutions. First proposed by Metropolis, simulated annealing draws on concepts from statistical mechanics. Tours which are longer than the current tour are accepted given a logarithmic probability which depends on a temperature parameter. First, the similarity to statistical mechanics will be discussed followed by Simulated Annealing's application to combinatorial optimization.

Simulated annealing so titled because it is just that: a model of annealing. Annealing refers to the cooling of metal or glass in order to overcome regions of stress that can be formed if the material is cooled improperly. The material starts at a high temperature and this temperature is cooled according to a specific schedule. This is done to allow the molecules to lose energy gradually, which leaves a structure stable. Conversely, if the material is cooled quickly (or quenched) it will "freeze" in an unstable, high energy state.

An initial temperature is stated and decreased as the number of iterations is increased. This decreases the probability of accepting an inferior solution as time goes on, causing the algorithm to settle in a local minima, which is hopefully (because of sometimes accepting a poorer tour) a global minima. One of the drawbacks to simulated annealing is the large number of iterations that is required to arrive at a minima. This is because initially many inferior solutions are accepted, and simulated annealing thoroughly searches that the entire space of solutions to arrive at its solution.

When applied to the traveling salesman problem, simulated annealing often takes the following form. An initial temperature level is defined. This temperature is lowered by some multiplicative factor during each iteration until a minimum temperature is reached. Decreasing the temperature decreases the probability of accepting an inferior solution. In this way many solutions are investigated initially in the search but as the search progresses fewer and fewer inferior solutions are accepted and the solution converges to a local minima which is hoped to be global.

Of these aforementioned local search techniques a common factor is that they all focus on the search

heuristic or algorithm as an area to improve upon. A recent development in computing which may prove useful in solving the traveling salesman problem is neural networks. A neural network works on the premise that a set of neurons can provide a solution to a problem based on analog, rather than digital, information. A neural net simulation programs an analog circuit to to run on a digital machine processor. Because of this hardware restriction, neural networks have not been able to solve TSP's larger than 30 cities. However, new processors have been designed specifically to handle neural network computations. These may provide better solutions on larger problems. in the future.

3. Jiggling Algorithm

All of the previously discussed methods of obtaining a solution tour rely on changing the search heuristic. A different approach would be to define a different neighborhood. Here it is proposed to perturb the actual problem, then apply a standard heuristic such as the space filling curve in an attempt to obtain more information about the problem and make the search more efficient. This novel neighborhood definition is based on the fact that a heuristic is simply a mapping from a problem to a solution. Knowing that a heuristic problem pair (h,p) is an encoding of a solution, a subset of the solution space can be generated by a set of heuristics, i.e. the heuristics generate a subset of solutions. Similarly, a subset of solutions may be generated by the application of a single heuristic to perturbed versions of the original problem. This means a subset of solutions can be generated by perturbing the original problem data (e.g. by changing the locations of the cities in a planar TSP, processing times in a job scheduling problem, etc.). Solution subsets may also be generated by simultaneously varying heuristics and problems (Storer, Vaccari & Wu 1989).

When local search is based on problem perturbation neighborhoods, it is necessary to evaluate the objective function (tour length) using the original problem data. A neighboring solution sequence is generated by applying the algorithm to the perturbed problem, and then the tour is evaluated using the original locations of the cities. Hence, the term problem search.

For the traveling salesman problem, such a search could be most easily accomplished by changing the locations of cities. Possible methods including rotating, shrinking and jiggling the city locations. Rotating the cities requires that all of the cities be rotated a given amount about the center point of the unit square. This maintains their relative positions, and allows the cities to be repositioned within the space filling curve. Rotating is an attempt to avoid the problems that occur when two cities are very close, yet separated by an SFC boundary. To overcome this, a new problem space is created by rotating. When the same space filling curve is reapplied, a different ordering of cities may be returned. The rotation angle thus defines a single dimensional search space.

A second method of problem perturbations is shrinkage. This requires decreasing the size of the problem, and applying the original space filling curve to obtain a tour which is different from the tour which is created on the full sized problem.

Both of these methods have shown encouraging results (Storer 1987), but neither gives a near optimal arrangement of cities. A new transform idea is that of jiggling. Jiggling is a tour improvement procedure which allows each city to move a random amount. The space filling curve is then reapplied to check if such a perturbation creates an improved tour. It seems reasonable that an optimal configuration should not be too far from the original configuration. So, there should be an optimal tour configuration of cities that resembles the original locations of the cities. This is due to the fact that the space filling curve preserves nearness and gives reasonably good tours (Storer 1987). The result is an entirely new way to define a neighborhood of solutions.

The evaluation considers the original city locations, but evaluates any possible change in the tour to see if the new tour is better than the incumbent one. The jiggling procedure is a variation of a 3-opt

neighborhood search, but it is an informed variation. This is due to the nature of the swap; when cities are jiggled one at a time, the problem is changed only slightly. In this heuristic, once the city location is perturbed, its jiggled location is arranged according to the space filling curve heuristic. If the tour changes, the tour length is evaluated to see if an improvement occurs. This heuristic is intuitively good because it is fast (only a single SFC value is calculated), and a high quality neighborhood of solutions (problem space rather than solution space) can be explored.

The actual heuristic is as follows:

1. Jiggle the location of a randomly selected city. The distance to jiggle is chosen randomly from a uniform distribution on both the X and Y coordinate, subject to the constraint that the city remains inside the unit square. The direction to jiggle is also chosen randomly.
2. Reorder the city in the space filling curve heuristic unit interval
3. Evaluate the change in tour length.
4. If the new solution is an improvement, accept the new tour.

An important consideration of this algorithm is computational speed. Often, a tradeoff occurs when more CPU time is required for each iteration. Hopefully, an increased processing time will be justified by better solutions. The important difference between problem space perturbation heuristics such as rotating and shrinking, and jiggling is that the former require the entire space filing curve heuristic to be run for all of the cities, while jiggling only needs an individual SFC value for the jiggled city. Because of this, jiggling is *much* faster.

Within each iteration, there are several important parameters. The first is the distance to jiggle. Empirical results indicate that a good way of selecting a distance is by choosing a random distance from a uniform distribution. It is necessary to find the best distance to jiggle so that the perturbations are large enough to generate new solutions, but not so large as to generate poor solutions.

One of the niceties of jiggling is that it is a local search method, and is therefore easily transformable to handle advanced search techniques such as simulated annealing. Standard simulated annealing can be used to occasionally accept inferior solutions. Or, a

temperature scheme can be used to change the jiggling parameter.

One potential problem with the jiggling heuristic is that it may yield the same solution before and after the jiggle. The chance of this occurring increases for smaller jiggle sizes. As mentioned above, it is desirable to find a good jiggle distance to overcome this problem. But this phenomena can also be used advantageously. When the tour is the same, the heuristic can be modified to accept the new location if it is closer to its original location and the tour is unchanged. Termed pseudo-annealing this technique helps prevent the problem from diverging from the original, and at the same time continues to progress through the search space even though the same solution is produced.

Another method that has given good results involves changing the perturbation size. Originally, the length of the jiggle was chosen from a uniform distribution with constant range. However, it has been found that by varying this jiggle length, the size of the search neighborhood is changed, thus allowing a more diverse search. This variation was implemented by geometrically increasing the jiggle size to a given limit and then returning to the original size and starting

again. This "rippling" effect allows the neighborhood size to expand and contract, instead of fixing it constant. For many iterations, this rippling of the jiggle size lets certain small perturbations balance against larger ones, and seems to give better results.

4. Experimentation

The generic jiggling algorithm was implemented in a FORTRAN77 program on a SUN 4/280 computer. The program works as follows:

1. The problem is created. N cities are randomly generated on a unit square.

2. The space filling curve values for each city are found and sorted, and an initial tour is created. Forward and Backward pointer are assigned to all cities so that the tour may be traversed in either direction.

3. The jiggling heuristic is implemented for a specified number of iterations.

- a. City B is randomly chosen as the city to jiggle. A precedes B, while C follows B in the current tour.

- b. A new SFC value is found for city B.

- c. Depending on the value found in b. the new position in the tour is found by either searching forward or backward in the tour using the pointers. This enhancement improves the computational speed of the algorithm.

- d. The difference between the new and current tour is calculated. If the difference is positive, the new tour is automatically accepted. Otherwise, the pseudo

annealing criteria is checked to determine if the tour is accepted.

4 The converging 2-opt procedure is applied to the tour resulting from the jiggling heuristic.

Initially, it was necessary to specify many of the parameters involved with the search. Using just a straight forward hillclimbing version of jiggling, the fundamental jiggling parameters were examined. This was done by ignoring the running time i.e. number of iterations. Allowing the jiggling heuristic to run for up to one million iterations on problems of size 1000 to 10,000 cities, it was found that the best results were obtained when the jiggle distance was chosen from a uniform distribution having a range parameter of 0.02 to 0.03. Because it worked better for all of the larger problems, the value of .02 was chosen as the base value for use during further experimentation.

The next step was to implement a simulated annealing temperature scheme on the jiggling heuristic. A standard scheme was used to vary the probability of accepting an inferior tour; an initial temperature was geometrically decreased during each iteration. However, the results of the simulated annealing were not as good as expected. Sometimes, simulated

annealing versions of the jiggling heuristic provided a better solution than straight hillclimbing, but usually, this was not the case. It was found that using a scheme which only allowed a few initial inferior solutions to be accepted worked better than a scheme which accepted many inferior solutions. In this latter case at least five times as many iterations were required to obtain the similar results. And although it was tried with each additional variation, simulated annealing did not prove to be worth the large additional computational time that was required. This is because the jiggling algorithm converges very quickly initially. This is the important constraint on the jiggling heuristic. Initially the tour is reduced rapidly, but as the tour converges towards a local optimum, the jiggling heuristic's ability to improve the tour decreases. This can be measured by finding the ratio of iterations that improve the tour as compared to the number of iterations completed. This "slope" can be seen in figure 1, as the slope is initially much greater than later stages of the heuristic. Using the 1000 city problem as an example, the tour is reduced from about 30 to about 27.5 in the first 10,000 iterations. In the next 20,000 iterations, that value is only reduced to about 27.2, and it takes an additional

TOUR LENGTH v ITERATIONS

for 1000 cities

33

tour length

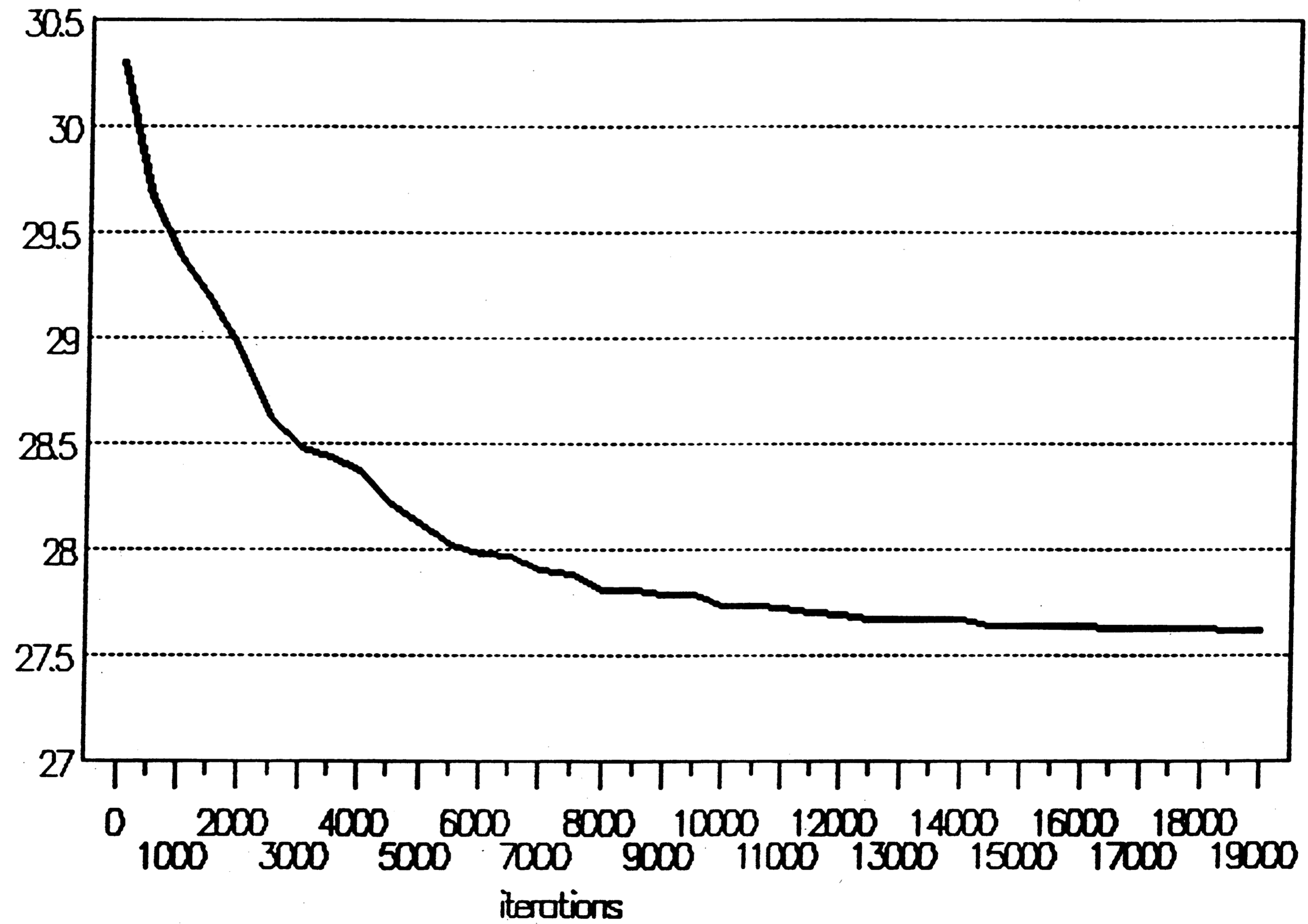


Figure 1

70,000 iterations to reduce the tour to 27.0. Figure 1 is for the 1000 city problem, but the curve is similar for all size problems. Because of this property, simulated annealing did not have the anticipated results.

After finding simulated annealing to be disappointing, another modification using the idea of accepting non-superior tours was implemented. As previously discussed, pseudo-annealing is a variation which accepts a new solution if the tour doesn't change, but the location of the jiggled city is closer to the original. This pseudo-annealing worked well and was incorporated into the foundation of the jiggling algorithm.

At this point, the jiggling heuristic, i.e. with pseudo-annealing and a fixed jiggle distance parameter of .02 gave results of about 15% above the expected optimal. As previously defined, the expected optimal for TSP's uniformly distributed on a unit square is calculated as $.765 \cdot \sqrt{N}$ (Beadwood et al). To further improve on this value, an alternate method in conjunction with the jiggling was tested.

In this new method, the jiggling algorithm was followed by a 2-opt neighborhood search. This procedure is intuitively good because jiggling is a 3-opt

procedure, which searches different spaces than a 2-opt. Thus, jiggling was followed by modified 2-opt in a composite procedure. Not surprisingly, the modified 2-opt improved the solution, but only a small amount. The modified 2-opt procedure was repeated a number of times, but the results did not justify the extra time required. Next, the modified 2-opt was repeated a number of times with varying modifying factors. That is, once jiggling is completed, start with a modifying factor of 100, once this modified 2-opt is completed, continue with a factor of 50, then 20 and so on until the modifying factor is 3. This "converging" 2-opt procedure was found to give good results when preceded by jiggling. It was necessary to determine if these results were due to the power of converging 2-opt or to jiggling. The converging 2-opt procedure was applied to initial tours that had not undergone the jiggling algorithm, and the results were not as good as those of jiggling. Alone, jiggling and the converging 2-opt procedure give acceptable results, but when combined the resulting tours are much better. An explanation for this is that two different neighborhoods are searched. A 3-opt neighborhood by jiggling and a 2-opt neighborhood by the converging 2-opt. Once the

3-opt neighborhood search is completed, the converging 2-opt is then able to search different neighborhoods as the search continues. By allowing the modifying factor to be reduced, the search neighborhood size is reduced as the tour length progresses towards a minima. A "diverging" 2-opt method was tried, i.e. modifying factor increased from 3 to 5 and so on up to 100, but results were poor because the search initially exhausted the 2-opt neighborhood, and the later searches were impractical. The results of the various 2-opts are shown in table 2 and figure 2. It can be seen of the 2-opts, the converging 2-opt is clearly the best of those tried.

The converging 2-opt procedure was appended to jiggling heuristic. The converging 2-opt could not be modified, so attention was once again focused on the jiggling procedure. As previously discussed, the jiggling algorithm is productive initially but slows considerably as the iterations increase and the tour length decreases. Would there be a way to extend the length of time that jiggling is productive? Many variations of the jiggling heuristic were tried. Examples of modifications that were tried include: "Always jiggle cities towards the center", "always jiggle away from the center", "jiggle

2-OPT COMPARISON

CITIES	EXP OPTM	MOD 2OPT 1	MOD 2OPT 3	STD 2OPT	excess	CONVG 2OPT	excess
1000	24.1914	27.2058	27.9032	28.1345	1.1630	27.1564	1.1226
2000	34.2118	38.7731	39.2887	41.0061	1.1986	38.1951	1.1164
3000	41.9008	47.2285	48.6136	49.5354	1.1822	46.9711	1.1210
4000	48.3828	55.2967	56.3268	57.7908	1.1944	53.6810	1.1095
5000	54.0937	61.4308	63.1885	64.3253	1.1891	60.0722	1.1105
6000	59.2566	67.4225	69.0780	70.6917	1.1930	65.9709	1.1133
7000	64.0045	73.1543	74.3702	76.3264	1.1925	70.5849	1.1028
8000	68.4237	77.6620	79.6073	82.1273	1.2003	76.2693	1.1147
9000	72.5743	82.5147	84.3945	86.7181	1.1949	80.6148	1.1108
10000	76.5000	87.3221	88.7986	91.3656	1.1943	84.7682	1.1081

37

This table show the comparison of 2opts to the expected optimal.

Column 3 is a modified 2opt with a modifying factor of 1.

Column 4 is a modified 2opt with a modifying factor of 3.

Column 5 is a standard 2opt.

Column 6 is the amount above optimal for the standard 2opt.

Column 7 is a Converging 2opt alone.

Column 8 is the amount above optimal for the converging 2opt.

Table 2

COMPARISON OF DIFFERENT 2-OPTS

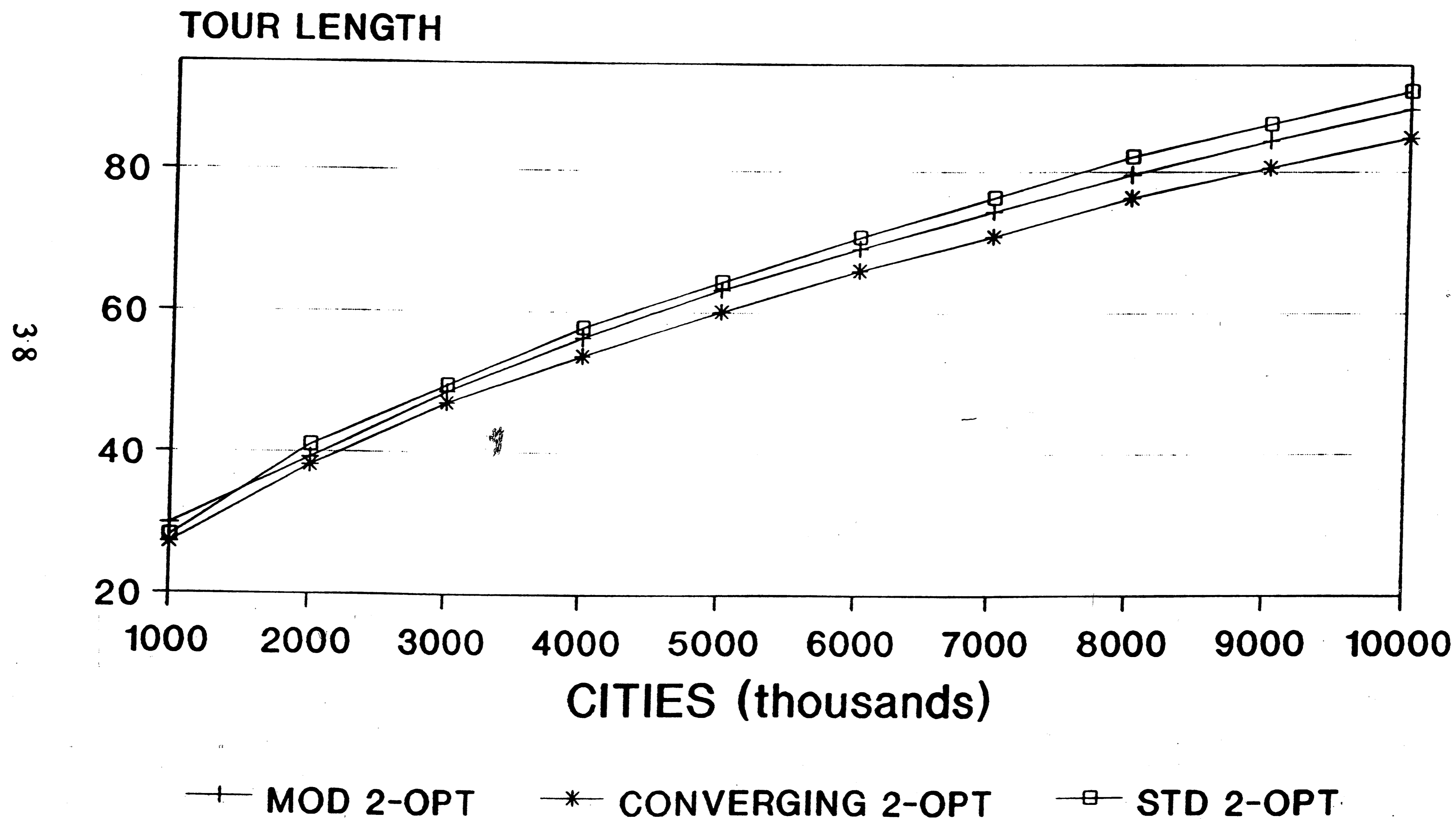


Figure 2

left or right but not up or down." None of these variations proved to work better than the original method.

Variations in the length of jiggle were also tried. A simulated annealing type of cooling schedule was used on the jiggle distance. Starting with an initial jiggle length of 0.03, decrease this value was decreased each iteration by a factor of 0.99. This heuristic did not prove to be useful. Neither did the opposite, increasing the jiggle length each iteration by a factor of 1.001. However, the rippling method discussed earlier proved to be very useful. Starting with a jiggle distance of 0.02, increase this value during each iteration until a maximum jiggle distance is reached and then return to the original value of 0.02. The tours which resulted were in almost all cases better than the tour which resulted with constant jiggle sizes. This is because this rippling jiggle size variation allows the 3-opt neighborhood search space to vary as the tour is decreased. The space varies from a distance of .02 to the maximum jiggle distance (MJD). This MJD is a factor of the size of the problem. For problems with 1000 cities a good MJD is .04 while a problem with 10,000 cities may have a good MJD of .065.

Finally, the number of iterations for jiggling and the modifying factors of the converging two-opt procedure were tied to the size of the problem. In order to meet all of the previously mentioned criteria (especially allowing the algorithm to be productive for as long as possible while not allowing too many unproductive iterations), the number of iterations was set at twenty times the number of cities in the problem. In order to provide an acceptable search neighborhood for the converging two-opt the modifying factors were set in accordance with the problem size. The first modifying factor is set at $.1*N$, the second at $.05*N$, and continues to a final value of $.003*N$. This makes sure that larger neighborhoods are searched in larger problems, and time is not wasted searching large neighborhoods in smaller problems. For example, the first modifying factor on the 1000 city problem is 100, while on the 10,000 city problem the first modifying factor is 1000. These different factors give similar neighborhood sizes for their respective problems.

The final jiggling algorithm contains the following refinements:

1. The number jiggle iterations equals 20 times the number of cities

2. Pseudo-annealing to preserve the original problem

3. Rippling of the jiggle distance to provide a changing search neighborhood size.

4. A converging two-opt procedure which converges at a rate dependent on the size of the problem.

Using the aforementioned algorithm to find the final tours, the best and average results for this heuristic are given in table 3. These results are encouraging as they compare favorably to three-opt procedures, and comparably to two-opt procedures of a similar computational time length. A comparison of CPU times is displayed in table 4.

FINAL RESULTS

CITIES	EXP OPTM	JIGGLING ALONE	std dev	CONV 2OPT ADDED	std dev	BEST RESULT	excess
1000	24.1914	27.3556	0.265	26.7883	0.316	26.1054	1.0791
2000	34.2118	38.7940	0.254	38.1076	0.374	37.3356	1.0913
3000	41.9008	47.8410	0.677	46.1228	0.322	45.6813	1.0902
4000	48.3828	55.3714	0.331	53.3122	0.144	53.1786	1.0991
5000	54.0937	61.8029	0.341	59.7094	0.381	59.0660	1.0919
6000	59.2566	67.0036	0.297	64.8605	0.238	63.4971	1.0716
7000	64.0045	72.2370	0.420	70.2415	0.287	69.9470	1.0928
8000	68.4237	78.6350	0.698	75.7291	0.314	75.3243	1.1009
9000	72.5743	82.7017	0.401	79.6576	0.336	79.1819	1.0910
10000	76.5000	86.9423	0.321	83.8537	0.286	83.6009	1.0928

42

THIS TABLE SHOW THE AVERAGE RESULTS OF JIGGLING ALONE
AND THE COMPLETE JIGGLING HEURISTIC INCLUDING
THE ADDITION OF THE CONVERGING TWO OPT.

Column 3 contains the results of the average of at least 8 problems of the
jiggling heuristic before the converging 2 opt was added.

Column 4 is the standard deviation of the jiggling alone.

Column 5 is the average of at least 8 problems of the complete heuristic.

Column 6 is the standard deviation of the complete heuristic.

Column 7 is the single best result for the given size problem.

Column 8 is the amount above optimal of the best solution.

Table 3

CPU TIME COMPARISON

CITIES	JIGGLES ALONE	CONV 2OPT ADDED	TOTAL JIGGLING ALGORITHM	MOD 2OPT
1000	28.83	4.33	33.16	11.67
2000	62.83	18.80	81.63	53.00
3000	89.50	49.00	138.50	119.00
4000	122.80	104.20	227.00	234.00
5000	147.73	153.00	300.73	310.00
6000	180.33	227.00	407.33	*
7000	209.75	330.00	539.75	*
8000	241.37	324.00	565.37	*
9000	267.46	386.00	653.46	*
10000	282.00	578.00	860.00	*

* Times are always larger than Jiggles Alone.
 AVERAGE TIMES ARE IN UNITS OF CPU SECONDS

Column 2 is Jiggling part of complete heuristic.
 Column 3 is Converging 2opt part of complete heuristic.
 Column 4 is Total heuristic time.
 Column 5 is Time for Converging 2opt from initial tour.

Table 4

5. Summary

The jiggling procedure discussed in this thesis works as well as it does because it is an informed three-opt neighborhood definition. This novel definition provides a search space that is different from those which are commonly explored.

The jiggling heuristic improves the initial tour about halfway from the space filling curve to the expected optimal. Figure 3 is a good example of this. As compared to some standard procedures, the problem space perturbation heuristic of jiggling alone seems to work comparably to a modified 2-opt procedure. The addition of a converging 2-opt procedure to the jiggling algorithm decreases the tour in a relatively short period of time.

The final analysis of the complete heuristic indicates the following. The results of the jiggling part alone compare favorably to a standard 2-opt or 3-opt search. The results of the complete algorithm, with the jiggling followed by converging 2-opt, are more favorable than those of a single neighborhood search, or a combined search of similar neighborhoods. This comparison can be seen in table 5.

Further experimentation may be needed to improve the jiggling heuristic to a point where it provides near optimal solutions. The area which might be likely to yield the greatest improvement is increasing the length of time that jiggling is productive. This will provide a better neighborhood which can be search efficiently for a longer period of time.

For large problems, the jiggling heuristic has given some very encouraging initial results. The 2-opt was once considered the best solution space, but it has since been improved on with 3-opt and the Lin-Kernighan search space. Jiggling is a new neighborhood definition which has great potential to be similarly improved upon.

COMPARISON OF PROCEDURES

CITIES	EXP OPTM	BEST JIGGLING	CONVER 2OPT	STD 2OPT	STD 3OPT
1000	24.1914	26.1054	27.1564	28.1345	29.5091
2000	34.2118	37.3356	38.1951	41.0061	42.3733
3000	41.9008	46.0253	46.9711	49.5354	52.0817
4000	48.3828	53.1786	53.6810	57.7908	59.8123
5000	54.0937	59.3685	60.0722	64.3253	67.3416
6000	59.2566	63.4971	65.9709	70.6917	74.1692
7000	64.0045	69.9552	70.5849	76.3264	79.7335
8000	68.4237	75.3243	76.5849	82.1273	85.3647
9000	72.5743	79.1819	80.6148	86.7181	90.6947
10000	76.5000	83.6009	84.7682	91.3656	95.6562

This table shows the results of the jiggling algorithm compared to other procedures.

Table 5

JIGGLING ALGORITHM VS SFC TOUR

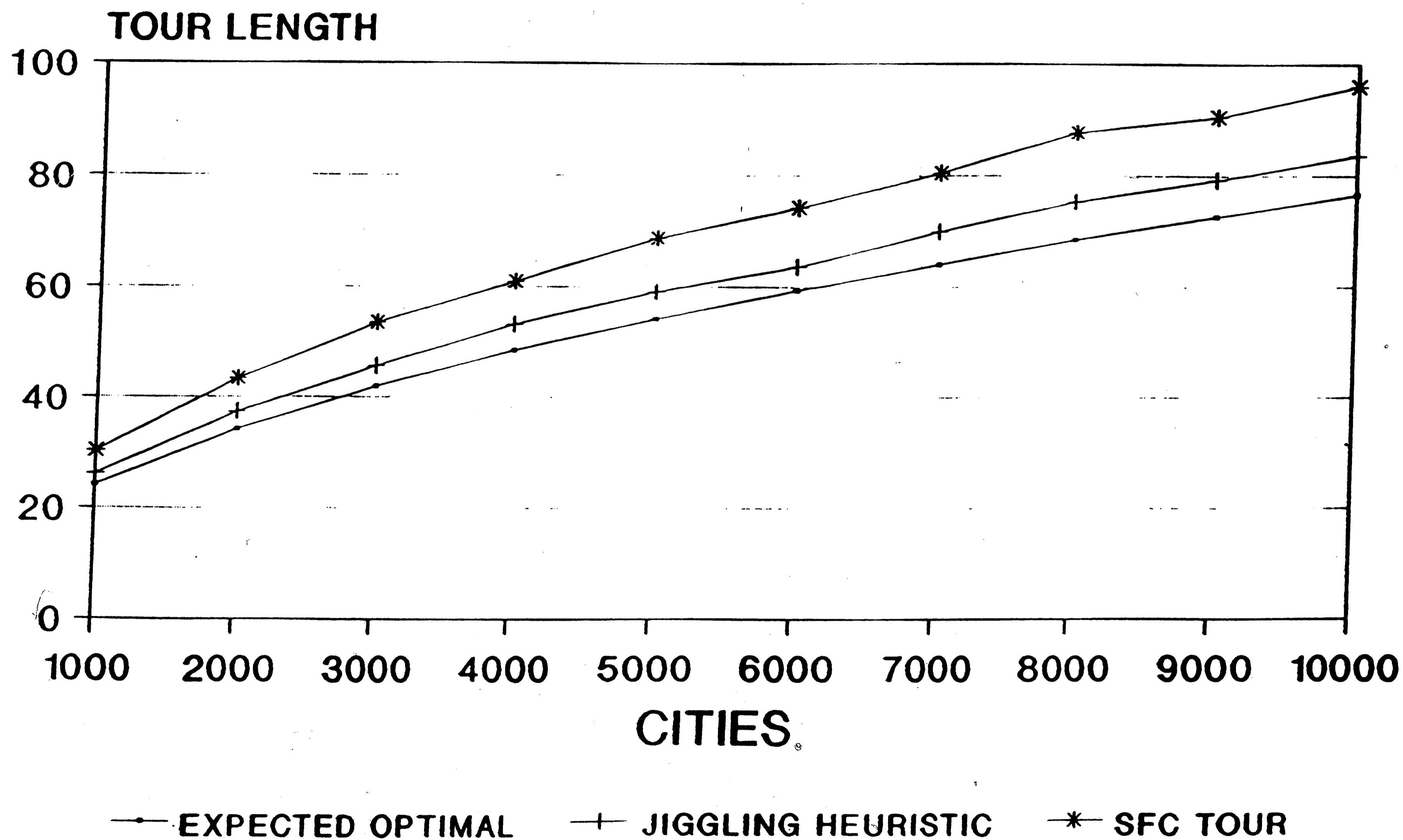


Figure 3

47

JIGGLING ALGORITHM VS CONVERGING 2OPT

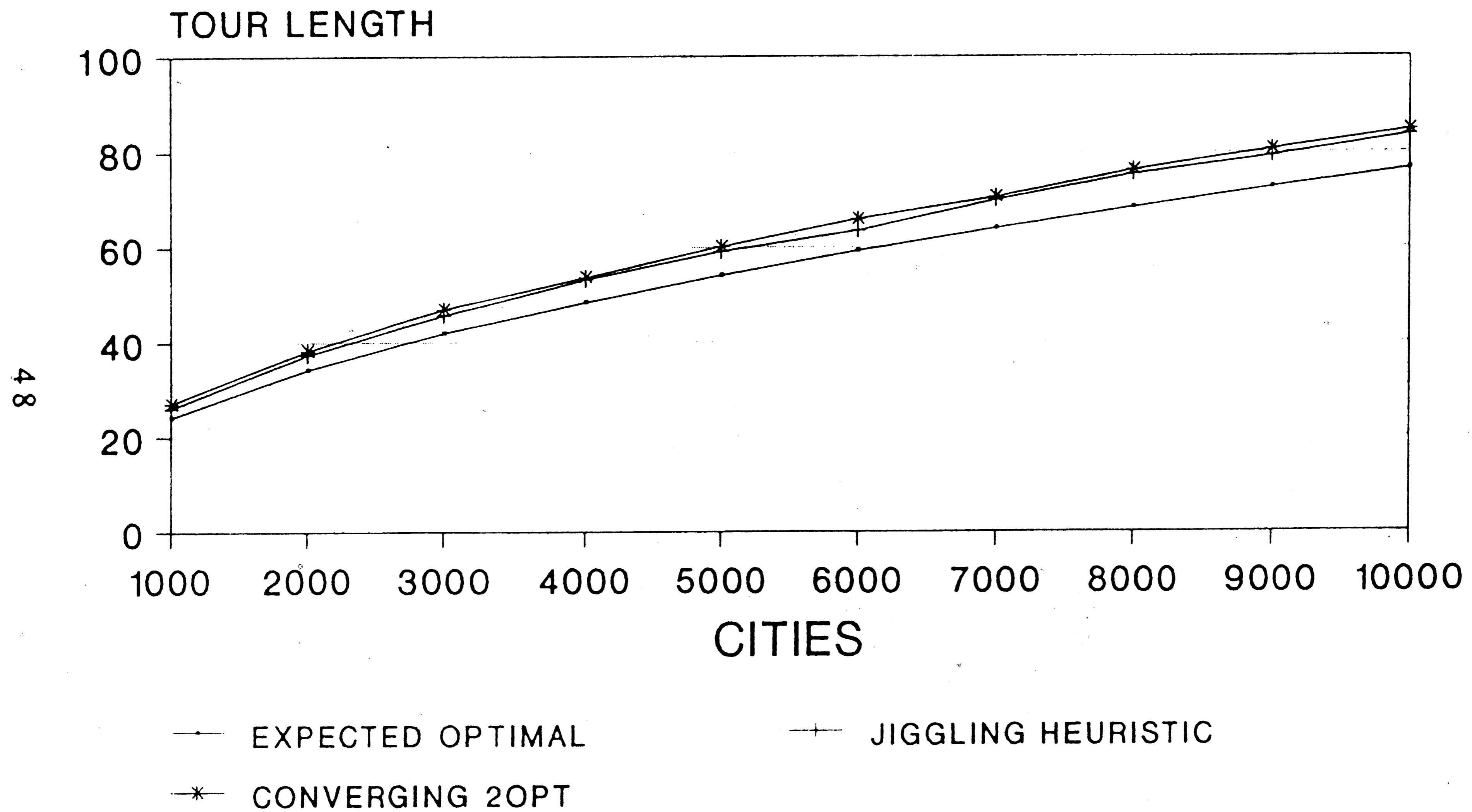


Figure 4

REFERENCES

Aarts, E.H.L. and Korst, J.H.M. "Boltzman Machines and Their Applications" Parallel Arch & Languages Europe (Spring 1987), pp. 34-50.

Bartholdi, J.J. and Platzman, L.K., "An $O(N \log N)$ Planar Traveling Salesman Heuristic based on Spacefilling Curves" Operations Research Letters 1 (1982) , pp. 121-125.

Bartholdi, J.J. and Platzman, L.K., "Heuristics based on Spacefilling Curves for Combinatorial Problems in Euclidean Space" Management Science 34 (1985), pp. 291-305.

Bonomi, E.L. and Lutton, J.L. " The N-City Traveling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm" SIAM Review 26 (October 1984), pp. 551-568.

Davis, G.W., "Sensitivity Analysis in Neural Net Solution" IEEE Transactions on Systems, Man and Cybernetics 19 (1989), pp.1078-1082.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP Completeness*. San Francisco: Freeman Press, 1979.

Golden, B., Bodin, L. Doyle, T. and Steward, W. Jr., "Approximate Traveling Salesman Algorithms" *Operation Research* 28 (May 1980), pp. 694-711.

Glover, F., "Tabu Search - Part I" *ORSA Journal of Computing* 1 (1989), pp 190-206.

Helbig Hansen, K. and Kraup, J., "Improvements of the Held- Karp Algorithm for the Symmetric Traveling Salesman Problem" *Mathematical Programming* 7 (1974), pp. 87-96.

Held, M., Hoffman, A.J., Johnson, E.L. and Wolfe, P., "Aspects of the Traveling Salesman Problem" *IBM Journal of Research and Development* 28 (July 1984), pp.476-486.

Held, M. and Karp, R.M., "The traveling-salesman problem and minimum spanning trees" *Operations Research* 18 (1970), pp. 6-25.

Held, M. and Karp, R.M., "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical Programming I* (1971), pp. 6-25.

Hillman, A.P., Alexanderson, G.L., and Grassl, R.M.
Discrete and Combinatorial Mathematics. San Francisco:
Dellen Publishing Co., 1987.

Hopfield, J.J. and Tank, D.W., "'Neural' computation of
decisions in Optimization Problems" Biological Cybernetics
52 (1985), pp. 141-152.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and
Schevon, C., "Optimization by Simulated Annealing: An
experimental Evaluation, Part I (Graph Partitioning)"
Operations Research 37 (1989), pp 865-892.

Johnson, D.S., "Local Optimization and the Traveling
Salesman Problem" Proceedings from the 17th Colloquium
on Automata, Languages & Programming, Springer-Verlag
(1990), pp. 446-461.

Kirkpatrick, S., "Optimization by Simulated Annealing:
Quantitative Studies" Journal of Statistical Physics 34(1984),
pp.976-986.

Kirkpatrick, S. , Gelatt, C.D., and Vecchi, M.P.,
"Optimization by Simulated Annealing" Science 220 (13 May
1983), pp. 671-680.

Korf, R.E., "Real Time Heuristic Search" *Artificial Intelligence* 42 (March 1990), pp. 189-211.

Lin, S. and Kernighan, B.W., "An effective heuristic Algorithm for the Travelling Salesman Problem" *Operations Research* 21 (1973) pp. 498-516.

Litke, J.D. , "An Improved Solution to the Traveling Salesman Problem with Thousands of Nodes" *Communications of the ACM* 27 (1984), pp. 1227-1236.

Minieka, E. *Optimization Algorithms for Networks and Graphs*. New York: Marcel Dekker, Inc., 1978.

McLaughlin, M.P., "Simulated Annealing" *Dr. Dobbs Journal* (September 1989), pp. 26-37.

Papadimitiou, C.H. and Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice Hall, 1982.

Pearl, J., *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley Pub Co., 1984.

Sahni, S., *Concepts in Discrete Mathematics*. Fridley, MN: The Camelot Pub. Co., 1981.

Steele, J.M., Snyder, T.L., "Worst-case Growth Rates of Some Classical Problems of Combinatorial Optimization" *SIAM Journal on Computing* 18 (April 1989), pp. 278-287.

Storer, R.H., and Bringham, A. "Heuristics for the Planar Euclidean TSP Based on Space Filling Curves and Simulated Annealing" Lehigh Dept. of I.E. (1987).

Storer, R.H., Vaccari, R. and Wu, S.D., "New Search Spaces for Sequencing Problems" Lehigh Dept of I.E. (1989).

Wasserman, P.D., *Neural Computing: Theory and Practice*. New York: Van Nostrand Reinhold, 1989.

Vita

Andrew J. Dvorocsik was born in Kingston, NY, the son of Michael and Dorothy Dvorocsik. He graduated with honors from Kingston High School in 1985. In 1989, he received his Bachelors of Science degree in Industrial Engineering from Lehigh University.

He immediately entered graduate school at Lehigh where he taught Engineering Computations for one year before receiving his Masters of Science degree, also in Industrial Engineering with a concentration in Operations Research.