SOME COMPLEXITY RESULTS FOR THE TRAVELING SALESMAN PROBLEM*

Christos H. Papadimitriou & Kenneth Steiglitz
Princeton University

ABSTRACT

It is shown that, unless P=NP, local search algorithms for the Traveling Salesman Problem having polynomial time complexity per iteration will generate solutions arbitrarily far from the optimal. The Traveling Salesman Problem is also shown to be NP-Complete even if its instances are restricted to be realizable by a set of points on the Euclidean plane.

1. Introduction

The Traveling Salesman Problem (TSP) can be stated as follows : Given r cities and (r-1) r/2 nonnegative integers denoting the distances between all pairs of cities, we are required to find a tour, that is a closed path passing through each city exactly once, so that the total traversed distance is minimal. Despite the simplicity of its statement , the TSP is apparently a very hard problem and has attracted a large number of researchers. Although no efficient algorithm for its solution has been found (and no nontrivial lower bound of its complexity has been proved) a number of different lines of attack have been proposed . A class of heuristics known as Local Search Algorithms ([Li],[LK],[RS],[SW]) have been particularly successful in generating good solutions for large problems by a reasonable computational effort. A local search algorithm (to be more formally defined later) starts with an essentially random tour and, by searching a set of tours which are considered "neighbors" of the former , either finds a neighbor with improved cost and uses it as a new starting point or, if this is not possible , terminates . The solution generated by this technique is called a **local optimum.** Tours of minimum length are referred to as **global optima.** Local optima may or may not necessarily be global optima, depending on the particular neighborhood structure used by the algorithm. Local search algorithms generating only global optima are called **exact.**

We will be particularly interested in the complexity of the problem of searching the neighborhood of a tour in order either to find an improvement or show this tour to be a local optimum. By "complexity of local search" the above mentioned complexity is understood -and not the complexity of the whole algorithm ,which heavily depends on the number of iterations necessary. In particular we will examine the computational requirements of local

*This work was supported by NSF Grant GK-42048 , the U.S.Army Research Office-Durham under Grant DAHC04-75-G-0192, and an IBM Fellowship.

search algorithms for the TSP, when certain restrictions are imposed on the quality of the obtained local optima .

The notion of a Combinatorial Optimization Problem with a Numerical Input (COPNI) is introduced . This class , which appears to be a restriction of the Subset Problems discussed by [SWK] ,includes several well-known problems such as the TSP and instances of the problem of Job Scheduling with Deadlines (JSD). A particular COPNI is exhibited in which the minimal exact neighborhood although exponential in size , can be searched in linear time. This counterexample shows that the cardinality of the minimal exact neighborhood is not , as [WSB] suggest , a lower bound for the complexity of exact local search.

In fact , if the exact local search problem were of provably exponential complexity ,this would be a rather remarkable result,since exact local search for the TSP is one of those tasks that are made very easy if non-deterministic computations are allowed. In the light of this observation we can think of the question , whether exact local search for the TSP can be done in a polynomial amount of time per iteration , as a part of the presently unsettled P=NP question. In fact it is shown that ,unless P=NP, each iteration of an exact local search algorithm for the TSP requires more than a polynomial number of steps.

A stronger result is also shown along the same lines. It is proved that , if a local search algorithm requires only a polynomial amount of time per iteration , the local optima thus obtained can be arbitrarily far from the optimum, unless, of course , P=NP. The above result suggests that a large class of efficient heuristics ([Li],[LK],[RS],[SW]) yield local optima of no guaranteed accuracy whatsoever.

In Section 4 we report some further results concerning the complexity of the TSP. Two equally natural (and practical) versions of the TSP are shown to have essentially the same complexity. We also show that an important restriction of the TSP (the "Euclidean" TSP) is not much easier than the general problem.

2.<u>Combinatorial</u> <u>Optimization</u> <u>Problems</u> <u>With</u> <u>Numerical</u> <u>Input</u>.

The set of nonnnegative integers is denoted by Z^+. For $n \in Z^+$ by $\mathbf{n}$ we shall denote the set $\{1,2,\dots,n\}$.

<u>Def:</u> A Combinatorial Optimization Problem with Numerical Input (COPNI) is a pair (n,F) , where $n \in Z^+$ is the **size** of the problem and F, a subset of $2^{\mathbf{n}}$, is the set of **feasible solutions.** We will require that there exists at least one feasible solution and that no feasible solution is properly contained in another.

An **instance** of the COPNI (n,F) is a function (numerical input) $c : \mathbf{n} \to Z^+$. In order to solve an instance c of the COPNI (n,F) we are required to find a feasible solution $f \in F$ such that $c(f) = \sum_{j \in f} c(j)$ is minimal.

Note that the size of a COPNI , as defined above , is not necessarily equal to the length of a string in $(0,1)^*$ required to describe an instance of the problem . Also note that the feasibility of a solution is not affected by the numerical input. On the other hand the non-containment requirement for the feasible solutions can be easily seen to be equivalent to the condition , that for each $f \in F$ there exists an instance c for which f is uniquely optimal.

<u>Examples</u>.

The TSP with r cities is a COPNI with

$n = \binom{r}{2}$ and with F being the set of all possible tours represented as sets of r intercity links.

The problem of Job Scheduling with Deadlines (JSD) [Ka] is a COPNI. Here we have a set **n** of jobs and for each job $j \in$ **n** we have the deadline D_j and the execution time T_j. A subset f of **n** is feasible if all jobs in **n**-f can be executed on a single processor within their deadlines, and no subset of **n** properly containing **n**-f enjoys this property. In the case of JSD the values of c can be thought of as rewards obtained for executing a job within its deadline, and our goal is to minimize the rewards lost. It should be emphasized that, unlike the formulation in [Ka], the numbers $\{D_j\}$ and $\{T_j\}$ are **not** considered as a numerical input here.

The Steiner Tree Problem, the Max Flow Problem, the Minimal Spanning Tree Problem and many others can be formulated as COPNI's.

<u>Def:</u> A **neighbourhood structure** for the COPNI (n,F) is a function $N: F \to 2^F$.

Informally, N assigns to each feasible solution f its **neighbourhood** N(f). We will also informally describe a **local search algorithm** for the COPNI (n,F) and the neighbourhood structure N as a deterministic algorithm with input $(f_0;c)$, where $f_0 \in F$ and c is an instance of (n,F). The algorithm is described below in terms of the function IMPROVE (f,c) which, when invoked , returns some $s \in N(f)$ such that $c(s) < c(f)$, if such an s exists, and returns 'no' otherwise.

```
f:=f_0;
while IMPROVE (f,c)¬='no' do
  f:= IMPROVE (f,c);
return f
```

The output of this algorithm is called a **local optimum** wrt N for the instance c of (n,F) . The performance of a local search algorithm depends on the complexity of the function IMPROVE , the number of iterations (executions of the <u>while</u> loop) and the quality of the local optima. The neighbourhood structure affects all the above factors. In particular N is **exact** if all local optima wrt N are also global optima. For example, if N(f) =F for all $f \in F$, N is trivially exact.

The following characterization has been adapted from [SWK]:

<u>Theorem 1</u> In a COPNI (n,F) there exists a unique minimal exact neighborhood structure given by

$$\bar{N}(f) = \left\{ s \in F : \begin{array}{l} \text{for some instance } c \\ s \text{ is uniquely optimal} \\ \text{with } f \text{ second to optimal} \end{array} \right\}. //$$

The exact nature of the set $\bar{N}(s)$ for a tour s in the case of the TSP is not known. In fact it has been recently shown [Pa2] that, unless P=NP, no concise, algorithmic-oriented characterization of this set exists. However [WSB] have shown that for an r city TSP ,$\bar{N}$ consists of sets of cardinality at least $((r-2)/2)!$. They continue by arguing that the exponential size of $\bar{N}$ implies that exact local search for the TSP must be inefficient. The following fact demonstrates that this argument is not generally valid:

<u>Fact:</u> There exists a COPNI (n,F) and $f \in F$ such that $\bar{N}(f)$ is exponential in size but can be searched in O(n) time.

<u>Proof:</u> Consider the JSD with n odd, $D_i = (n-1)/2$ for $i=1,2,\ldots,n$, $T_1 = (n-1)/2$, and $T_j = 1, j=2,3,\ldots,n$.
The set of feasible solutions is

$F = \{f\} \cup F'$, where

$f = \{2,3,\ldots,n\}$ and

F' consists of all subsets of **n** of cardinality $(n+1)/2$ containing 1.

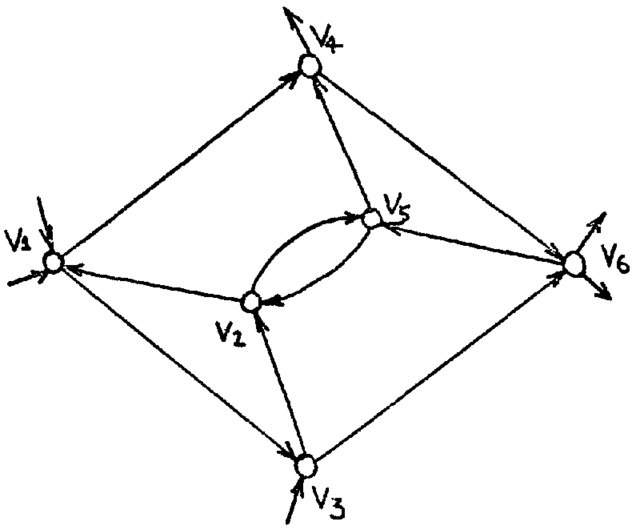
Consider any $s \in F'$. We can define an instance c_s as follows

$$c_s(j) = \begin{cases} (n-3)/2 & \text{if } j=1 \\ 0 & \text{if } j \neg= 1 \text{ and } j \in s \\ 1 & \text{otherwise.} \end{cases}$$

It can be easily verified that, for this instance, s is uniquely optimal (with cost $(n-3)/2$) with f second to optimal (cost $(n-1)/2$). Hence by Theorem 1, $s \in \bar{N}(f)$ and consequently $\bar{N}(f)=F'$. The cardinality of $\bar{N}(f)$ is approximately equal to $.8n^{-1/2}2^n$.

Yet for any instance c, $\bar{N}(f)$ can be searched in linear time. To see this, let t be the set of jobs in $\{2,3,\ldots n\}$ having the $(n-1)/2$ largest costs. The optimum is either f or $n-t$, depending on whether or not $\sum_{j\in t} c(j) < c(1)$. Consequently in order to search $\bar{N}(f)$ we only need to find the $(n-1)/2$ jobs in $\{2,3,\ldots,n\}$ having the largest cost, and compare the sum of their costs to $c(1)$. But this can be done in $O(n)$ time by using the median algorithm of [BFPRT].//

The idea behind this counterexample is that the minimal exact neighborhood is a data indeoendent set, whereas data can be used very efficiently in order to facilitate its search. As we will see in the next section there is little hope that something similar can be done in the case of the TSP.



3. The Complexity of Exact and Approximate Local Search.

For the purpose of relating the complexity of local search to the P=NP question, we now show certain related languages to be NP-Complete. We assume the existence of a function e mapping graphs, digraphs, paths, TSP tours and instances to strings in $\{0,1\}^*$. A wide variety of "reasonable" encodings would suffice for our purposes.

Def: Let V be the set of nodes of a graph (V,E) (resp. a digraph (V,E')) and let $(v_1,v_2,\ldots v_n)$ be a permutation of V such that (v_i,v_{i+1}) is an edge (resp. a directed edge) for $i=1,2,\ldots n-1$. If (v_n,v_1) is an edge (resp. directed edge) then $(v_1,\ldots v_n,v_1)$ is an undirected Hamiltonian circuit (UHC) (resp. directed Hamiltonian circuit (DHC)). Otherwise, if (v_n,v_1) is not an edge (resp. directed edge) then $(v_1,\ldots,v_n)$ is an undirected Hamiltonian path (UHP) (resp. directed Hamiltonian path (DHP)). Note that, by the above definition, no part of a Hamiltonian circuit is a Hamiltonian path.

In [Ka] the problems of determining whether a given graph (directed or undirected) has a Hamiltonian circuit are shown to be NP-Complete. We show that they remain NP-Complete even if a serious restriction is imposed on their domains. In particular one would expect that the search for a Hamiltonian circuit in a graph would be facilitated considerably, if we were given a Hamiltonian path . The next two Theorems suggest that this is not the case.

The restricted directed Hamiltonian circuit problem is the recognition problem for the following language:

$$\text{RDHC}=\{\, e(G);e(P) : \begin{array}{l}P \text{ is a DHP in } G\\ \text{and } G \text{ has a DHC}\end{array} \}$$

Theorem 2 RDHC is NP-Complete.

For the proof of the Theorem the following Lemma is needed:

Lemma Let the digraph H (shown in Fig. 1) be a subgraph of a digraph G, such that edges of $G-H$ enter H only at v_1 or v_3 and leave H at v_4 or v_6 only. Then, if G has a DHC C, one of the paths $(v_1,v_3,v_2,v_5,v_4,v_6)$ or $(v_3,v_6,v_5,v_2,v_1,v_4)$ is a part of C.

The proof of the Lemma is an exhaustive case by case analysis and is omitted.

Proof of Theorem 2 By reducing the DHC Problem to RDHC. Let $G=(V,E)$ be an instance of the DHC Problem. We will construct a digraph $G'=(V',E')$ with a DHP P, such that G' has a DHC iff G has a DHC.

We let $V=\{v^1,v^2,\ldots v^n\}$ and $V'=\{v_1^1,v_2^1,\ldots,v_6^1,v_1^2,\ldots,v_6^n\}$. For each $j\leq n$ we connect the nodes $\{v_1^j,v_2^j,\ldots,v_6^j\}$ as $\{v_1,v_2,\ldots,v_6\}$ are connected in H, and we call the resulting subgraph H^j. Moreover for each edge $(v^i,v^j)\in E$ we add the edge (v_6^i,v_1^j) to E'. We also add the edges (v_4^i,v_3^{i+1}), $i=1,2,\ldots n-1$ to E'.

Obviously G' has a DHP, namely $P=(v_3^1,v_6^1,v_5^1,v_2^1,v_1^1,v_4^1,v_3^2,v_6^2,\ldots,v_1^n,v_4^n)$. Moreover if the original graph G has a DHC $(w^1,w^2,w^3,\ldots,w^n,w^1)$, then the resulting graph G' also has a DHC, namely $(w_1^1,w_3^1,w_2^1,w_5^1,w_4^1,w_6^1,w_1^2,\ldots,w_6^n,w_1^1)$.

Conversely, suppose that G' has a DHC C. Suppose that for some i, C enters H^i at v_3^i. By the Lemma, $(v_3^i,v_6^i,v_5^i,v_2^i,v_1^i,v_4^i)$ is a part of C. Since v_3^{i+1} is the only node in $G'-H^i$ which succeeds v_4^i, it follows that C will enter H^{i+1} at v_3^{i+1}. Hence the same argument can be applied to H^{i+1}. Inductively, we can assume that C enters H^n at v_3^n. By the Lemma, $(v_3^n,v_6^n,v_5^n,v_2^n,v_1^n,v_4^n)$ will be a part of C. But there is no node in $G-H^n$, which succeeds v_4^n. Consequently C is not a DHC as supposed.

From the above contradiction we deduce, that for no $i\leq n$ will C enter H^i at v_3^i, and hence C is equal to $(w_1^1,w_3^1,w_2^1,w_5^1,\ldots w_6^n,w_1^1)$ for some DHC $(w_1,w_2,\ldots,w_n,w_1)$ of G. Consequently G has a DHC iff G' has a DHC, and the proof is completed. (The straight-forward verification of the facts that the problem is in NP and that the reduction is a polynomial-time one has been omitted).//

Similarly we define the restricted undirected Hamiltonian circuit problem to be the recognition problem of the language RUHC $=\{e(G);e(P):$ P is an UHP in G and G has an UHC. $\}$

Theorem 3 RUHC is NP-Complete.

Proof. By reducing the RDHC to it. The construction is identical to the one used in the proof of the NP-Completeness of the ordinary UHC problem ([Ka],[AHU]) . It is an elementary observation that the construction preserves the existence of a Hamiltonian path.//

An interesting side problem of the TSP is the following: Given an instance c and an edge (i,j), does (i,j) appear in some optimal tour? This problem is also NP-Complete . To show this, we define the language

M={e(c);e(i,j):the edge (i,j) does not appear in any optimal tour of the instance c of the TSP}.

Theorem 4 M is NP-Complete.

Proof. By reducing the RUHC to it. Let $(G;P)$ be an instance of the RUHC, where $P=(w_1,w_2,\ldots,w_n)$ is an UHP. Let c be an instance of the TSP such that $c(w_i,w_j)=2$ if (w_i,w_j) is not an edge of G, and $c(w_i,w_j)=1$ otherwise. If $(G,P)\in$RUHC, then G has an UHC and hence (w_1,w_n) (which, by definition of an UHP corresponds to a missing edge of G) will not appear in any optimal tour of c. Conversely, if (w_1,w_n) does not appear in any optimal tour of c, then the tour corresponding to P is suboptimal and hence G has an UHC. Consequently $(c,(i,j))\in M$ iff $(G,P)\in$RUHC.//

We now define the following language

$$L_0=\left\{e(c);e(f):\begin{matrix}\text{f is a suboptimal}\\ \text{tour for the instance c}\end{matrix}\right\}.$$

It can be argued that L_0 adequately captures the complexity per iteration of the exact local search problem for the TSP, since the recognition problem for L_0 can be solved by one call of the function IMPROVE (c,f) of any exact local search algorithm. Hence the following result suggests that exact local search for the TSP probably requires iterations of complexity more than polynomial:

Theorem 5 L_0 is NP-Complete.

Proof. By reducing RUHC to it. Let $(G=(V,E);P)$ be an instance of the RUHC problem. Let c be an instance of the TSP with $|V|$ cities, such that $c(v,u)=1$ if $(v,u)\in E$ and $c(v,u)=2$ otherwise. Let f be the tour corresponding to the path P. Then $(G,P)\in$RUHC iff $(c,f)\in L_0$.//

Let ϵ be any positive real number, and c an instance of the COPNI (n,F), with optimal feasible solution s . A feasible solution $f\in F$ is called **ϵ-approximate** [SG] if $(c(f)-c(s))/c(s) \leq \epsilon$. Otherwise f is called **ϵ-suboptimal**. In a similar way to L_0, the following language is defined for $\epsilon>0$:

$$L_\epsilon=\left\{e(c);e(f):\begin{matrix}\text{f is an }\epsilon\text{-suboptimal}\\ \text{tour for the instance c}\end{matrix}\right\}.$$

Theorem 6. L_ϵ is NP-Complete for all $\epsilon>0$.

Proof: Let $(G=(v,e),P)$ be an instance of the RUHC problem. Let c be the instance of the $|V|$-city TSP with $c(v,u)=1$ if $(v,u)\in E$ and $c(v,u)=2+\lceil|V|\epsilon\rceil$ otherwise. f is again the tour corresponding to P. It can be easily seen that $(G,P)\in$RUHC iff $(c,f)\in L_\epsilon$. //

Similarly to the exact case, the complexity of the recognition problem for L_ϵ is a lower bound for the complexity per iteration of any local search algorithm for the TSP, the local optima of which are guaranteed to be ϵ-approximate (since one call of the function IMPROVE of any such algorithm would solve the RUHC problem via the reduction outlined in the proof of Theorem 6). Hence Theorem 6 implies that, unless P=NP, all local search algorithms (such as the ones proposed by [Li],[LK],[RS],[SW]) whose iterations require only a polynomial amount of time, will yield local optima that can be arbitrarily far from the global optimum.

It should be emphasized that Theorem 6 and its implications are valid when no additional restrictions are imposed on the instances of the TSP considered. For example, if a "natural" constraint -the triangle inequality- holds among the intercity distances, [RSL] have shown that 1-approximate solutions can be obtained by algorithms (not necessarily iterative) of polynomial time complexity.

We also note that the methodology used in the proofs of this Section appears to be a fairly strong tool for understanding the nature (and limitations) of local search. For example we can construct an instance c of the TSP, in which two disjoint tours s and t satisfy $s\in N(t)$. This settles (for the directed case) a conjecture by [Li].

On the other hand by similar methods we can construct a family of pathological instances of the TSP, on which any local search algorithm should perform arbitrarily badly. These instances (defined on r cities) have (r/18)! tours that are (r/3)-optimal (in the terminology of [Li]), and a global optimum which is arbitrarily better. For the non-symmetric case these parameters can be improved to (r/6)! and r respectively.

4. The Complexity of the Euclidean TSP

Although the motivation for the TSP can probably be traced back to the Euclidean case (the cities are points on the two-dimensional Euclidean space with integer coordinates and the distances are the usual Euclidean metric) there is little known about the complexity of the Euclidean TSP. There is a general feeling in the literature (eg [LK]) that the Euclidean TSP is considerably easier than the general case, either because the heuristics seem to perform better or because special methods of attack are applicable. For example it is almost always easy in the Euclidean TSP to exhibit edges that are not contained in any optimal tour (namely the chords of the convex hull of the cities), whereas the same task seems to be considerably harder in the general case the same task seems to be considerably harder in the general case (see for example Theorem 4). Nevertheless in this Section it is shown essentially that the Euclidean TSP cannot be "much" easier than the general problem.

In fact we are dealing with two problems. The first, the tour-TSP, is the ordinary TSP. The other, the path-TSP, is the problem facing traveling salesmen who can start from any city and are not particularly interested in returning to the starting city of their tour. The path-TSP can be especially useful as a more precise model for some problems arising in applications, like the hole drilling problems [LK].

The following suggests that the computational requirements of these problems are closely related to each other:

Theorem 7 The problems tour-TSP and path-TSP reduce to each other in linear time.

Proof: Starting with the path-TSP, create a new city with equal distances from all other cities. An optimal tour in the resulting TSP corresponds in a natural way cities. An optimal tour in the resulting TSP corresponds in a natural way to an optimal path in the original.

For the opposite direction suppose that we have n cities $c_1,\dots,c_n$ and that k equals n times the largest distance between any two cities. Let d be the distance function (generalized to denote the length of paths and tours). We create a new copy c'_1 of c_1, and modify d as follows:

$d'(c_i,c_j)=d(c_i,c_j)$ if $i,j\neq 1$

$d'(c_1,c_j)=d(c_1,c_j)+2k$ for all j

$d'(c'_1,c_j)=d(c_1,c_j)+2k$ for all j

$d'(c_1,c'_1)=3k$

It is not hard to see that in the resulting instance all optimal paths must have c_1 and c'_1 as endpoints. Hence all such paths correspond to a tour in the original instance, and minimizing paths in the resulting TSP is equivalent to minimizing tours in the original.//

It is not clear though how the Euclidean cases of these problems relate to each other. Obviously the above reductions do not work. Of course one way to reduce the Euclidean cases to each other is to show that both are NP-Complete(Theorems 8 and 9).

We will now give a more precise definition of the Euclidean TSP. The cities can be given in terms of a list of pairs of integers denoting the coordinates with respect to some coordinate system. It is not clear what the distance matrix should be. If we take it to be the (infinite precision) real-valued Euclidean metric, it is a nontrivial task to show that the resulting problem is in NP, since there is no obvious upper bound for the precision required in order to compare the length of a tour or path with a given integer. In what follows we will assume that the elements of the distance matrix are the integral parts of this metric. Any desired precision can be thus obtained by increasing the scale accordingly.

Theorem 8: The Euclidean path-TSP is NP-Complete.

The proof of this Theorem is a reduction of the exact cover problem [Ka] to the Euclidean path-TSP. The construction is essentially an elaboration on the one used in [GJS] for the proof of the NP-Completeness of the planar Hamiltonian path problem for digraphs. Since the whole construction is rather complicated, it will not be detailed here (for the complete description of the proof see[Pa1]). By a similar construction we can show that the Euclidean version of the tour-TSP problem is also NP-Complete.

Theorem 9:* The Euclidean tour-TSP is NP-Complete .

The same technique can be used in order to prove that another restriction of the TSP, the "rectilinear" (or "Manhattan") TSP is also NP-Complete.

*Garey and Johnson [Jo] have independently reached the same conclusion.

5. References

[AHU] A.V.Aho, J.E.Hopcroft, J.D.Ullman The Design and Analysis of Computer Algorithms, Addison- Wesley [74]

[BFPRT] M.Blum, R.W.Floyd, V.R.Pratt, R.L.Rivest, R.E.Tarjan "Time bounds for selection", JCSS 7,4, pp 448-461 [72]

[GJS] M.R.Garey,D.S.Johnson,L.Stockmeyer "Some Simplified NP-Complete Problems", Proc. 6-th STOC, pp.47-63,[74].

[Jo] D.S.Johnson, private communication, October [75]

[Ka] R.M.Karp "Reducibility Among Combinatorial Problems", in: Complexity of Computer Computations, R.E.Miller and J.W.Thatcher eds Plenum Press, New York, pp85-103 [72]

[Li] S.Lin "Computer Solutions of the Traveling Salesman Problem", Bell System Technical Journal, 44, pp 2245-2269 [65]

[LK] S. Lin, B.W. Kernighan "An Effective Heuristic Algorithm for the Traveling Salesman Problem", OR, 21, 2, pp 498-516, [73]

[Pa1] C.H.Papadimitriou "The Euclidean Travelling Salesman Problem is NP-Complete", Princeton University CS TR No 189 [75]

[Pa2] C.H.Papadimitriou "On the Complexity of the Local Structure of Certain Convex Polytopes", Princeton University CS TR No 197 [75]

[RSL] D.J.Rosenkrantz, R.E.Stearns, P.M.Lewis "Approximate Algorithms for the Traveling Salesperson Problem", Proc. 15th SWAT Conf. pp33-428 [74]

[SG] S.Sahni,T.Gonzales "P-Complete Problems and Approximate Solutions " Proc. 15th SWAT Conf. pp.28-32 [74]

[SWK] S.L.Savage, P.Weiner, M.J.Krone "Convergent Local Search", RR#14, Yale

University Dept. of CS [73]

[RS] S.Reiter, G.S.Sherman "Discrete Optimizing", SIAM Journal,13, 3, pp864-889 [65]

[SW] K.Steiglitz,P.Weiner "Some Improved Algorithms for Computer Solution of the Traveling Salesman Problem", Proc. 6th Annual Allerton Conference on Circuit and Systems Theory, pp814-821, [68]

[WSB] P.Weiner, S.L.Savage, A.Bagchi "Neighborhood Search Algorithms for Finding Optimal Traveling Salesman Tours Must Be Inefficient", Proc. 5th ACM STOC [73]