SIAM J. COMPUT.
Vol. 6, No. 1, March 1977

ON THE COMPLEXITY OF LOCAL SEARCH FOR THE TRAVELING SALESMAN PROBLEM*

CHRISTOS H. PAPADIMITRIOU AND KENNETH STEIGLITZ†

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.

Key words. traveling salesman problem, local search, complexity, NP-complete

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 [5], [6], [11], [12] 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 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 [10], 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

* Received by the editors September 24, 1975.

† Department of Electrical Engineering, Princeton University, Princeton, New Jersey 08540. This work was supported by the National Science Foundation under Grant GK-42048 and by the U.S. Army Research Office—Durham under Contract DAHC04-69-C0012.

exponential in size, can be searched in linear time. This counterexample shows that the cardinality of the minimal exact neighborhood is not 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 nondeterministic computations are permitted. 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 [5], [6], [11], [12] yield local optima of no guaranteed accuracy whatsoever.

2. Combinatorial optimization problems with numerical input. The set of nonnegative integers is denoted by Z^+. For $n \in Z^+$ we shall denote by $\bar{n}$ the set $\{1, 2, \cdots, n\}$.

DEFINITION. A combinatorial optimization problem with numerical input (COPNI) is a pair (n, F), where $n \in Z^+$ is the *dimension* of the problem and F, a subset of $2^{\bar{n}}$, 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) $c: \bar{n} \to Z^+$. 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 feasibility of a solution is not affected by the numerical input. On the other hand the noncontainment 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.

There is an interesting geometric interpretation of COPNI's: every feasible solution in F corresponds (in the obvious manner) to a vertex of the n-dimensional hypercube. Hence the convex hull of these vertices is an equivalent representation of the COPNI. Since an instance of the COPNI is essentially a linear functional, it follows that solving an instance of a COPNI is equivalent to minimizing a linear functional over the vertices of a convex polytope. For a further discussion of this analogy, see [7].

Examples. 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) [4] is a COPNI. Here we have a set $\bar{n}$ of jobs and for each job $j \in \bar{n}$ we have the deadline D_j and the execution time T_j. A subset f of $\bar{n}$ is feasible if all jobs in $\bar{n} - f$ can be executed on a single processor within their deadlines, and no subset of $\bar{n}$ properly containing $\bar{n} - f$ 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 [4], the numbers $\{D_i\}$ and $\{T_i\}$ 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.

DEFINITION. A *neighborhood structure* for the COPNI (n, F) is a function $N: F \to 2^F$.

Informally, N assigns to each feasible solution f its *neighborhood* $N(f)$. We will also informally describe a *local search algorithm* for the COPNI (n, F) and the neighborhood 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 $(f, c) \neg =$ 'no' **do**
  $f :=$ IMPROVE (f, c);
return f

The output of this algorithm is called a *local optimum* with respect to 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 **while** loop) and the quality of the local optima. The neighborhood structure affects all the above factors. In particular N is *exact* if all local optima with respect to N are also global optima. For example, if $N(f) = F$ for all $f \in F$, N will be trivially exact.

The following characterization has been adapted from [10]:

THEOREM 1. *In a* COPNI (n, F) *there exists a unique minimal exact neighborhood structure given by*

$$\hat{N}(f) = \{s \in F: \text{ for some instance } c, s \text{ is uniquely optimal with } f \text{ second to optimal}\}$$

The exact nature of the map $\hat{N}$ for the case of the TSP is not known. In fact, the results in [7] suggest that there is no concise, algorithmic-oriented characterization of $\hat{N}$ for the TSP. However, the authors of [13] have shown that for an r city TSP, $\hat{N}$ consists of sets of cardinality at least $((r-2)/2)!$. They continue by arguing that the exponential size of $\hat{N}$ implies that exact enumerative local search for the TSP must be inefficient. The following fact demonstrates that this argument is not valid when nonenumerative (data-dependent) search is allowed:

FACT. *There exists a* COPNI (n, F) *and an* $f \in F$ *such that* $\hat{N}(f)$ *is exponential in size but can be searched in* $O(n)$ *time.*

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

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

where $f = \{2, 3, \cdots, n\}$ and

$$F' = \{s \text{ subset of } \bar{n}: 1 \in s \text{ and } |s| = (n+1)/2\}.$$

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, $s \in \hat{N}(f)$ and consequently $\hat{N}(f) = F'$. The cardinality of $\hat{N}(f)$ is approximately equal to $.8n^{-1/2}2^n$.

Yet for any instance c, $\hat{N}(f)$ can be searched in linear time. To see this, let t be the set of jobs in $\{2, 3, \cdots, n\}$ having the $(n-1)/2$ largest costs. The optimum is either f or $\bar{n} - t$, depending on whether or not $\sum_{j \in t} c(j) < c(1)$. Consequently in order to search $\hat{N}(f)$ we only need to find the $(n-1)/2$ jobs in $\{2, 3, \cdots, n\}$ 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 [2]. □

The idea behind this counterexample is that the minimal exact neighborhood is a data-independent 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.

DEFINITION. Let V be the set of nodes of a graph (V, E) (resp. a digraph (V, E')) and let $(v^1, v^2, \cdots, v^n)$ be a permutation of V such that (v^i, v^{i+1}) is an edge (resp. a directed edge) for $i = 1, 2, \cdots, n-1$. If (v^n, v^1) is an edge (resp. directed edge), then $(v_1, \cdots, v_n, v_1)$ 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 $(v^1, \cdots, v^n)$ 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 [4] 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:

$$\text{RDHC} = \{e(G); e(P): P \text{ is a DHP in } G \text{ and } G \text{ has a DHC}\}.$$

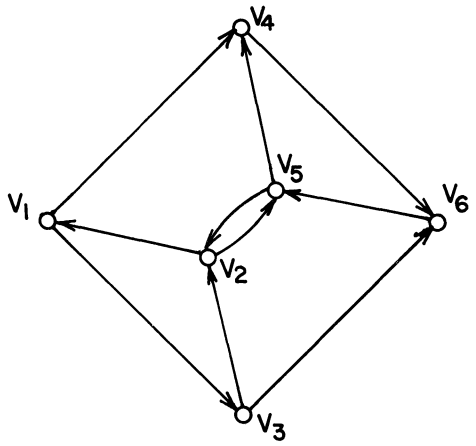

FIG. 1 *The digraph H*

THEOREM 2. RDHC *is* NP-*complete.*

For the proof of Theorem 2, 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.*

Proof of Lemma. Let C enter H at v_1. Then for some node u of $G-H$ one of the following six paths is a part of C:

1. (v_1, v_4, u),
2. (v_1, v_4, v_6, u),
3. (v_1, v_3, v_6, u),
4. $(v_1, v_3, v_2, v_5, v_4, u)$,
5. $(v_1, v_3, v_6, v_5, v_4, u)$,
6. $(v_1, v_3, v_2, v_5, v_4, v_6, u)$.

In the first five cases it can be easily verified that there is no way for C to pass through the unvisited nodes of H, contrary to our assumption that C is Hamiltonian. Consequently if C enters H at v_1, $(v_1, v_3, v_2, v_5, v_4, v_6)$ is a part of C. If C enters H at v_3, then, again, for some node u of $G-H$ one of the following seven paths is a part of C:

1. (v_3, v_2, v_1, v_4, u),
2. $(v_3, v_2, v_1, v_4, v_6, u)$,
3. (v_3, v_2, v_5, v_4, u),
4. $(v_3, v_2, v_5, v_4, v_6, u)$,
5. (v_3, v_6, u),
6. (v_3, v_6, v_5, v_4, u),
7. $(v_3, v_6, v_5, v_2, v_1, v_4, u)$.

Again, if one of the first six paths is indeed a part of C, C cannot visit the remaining nodes of H. Hence if C enters H at v_3, $(v_3, v_6, v_5, v_2, v_1, v_4)$ is a part of C, which completes the proof of the lemma. □

Proof of Theorem 2. We reduce 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 $V=\{v^1, v^2, \cdots, v^n\}$ and $V'=\{v_1^1, v_2^1, \cdots, v_6^1, v_1^2, \cdots, v_6^n\}$. For each $j \leqq n$ we connect the nodes $\{v_1^j, v_2^j, \cdots, v_6^j\}$ as $\{v_1, v_2, \cdots, v_6\}$ 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}), $i=1, 2, \cdots, n-1$, to E'.

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

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 $i \leqq n$ will C enter H^i at v_3^i, and hence C is equal to $(w_1^1, w_3^1, w_2^1, w_5^1, \cdots, w_6^n, w_1^1)$ for some DHC $(w_1, w_2, \cdots, w_n, w_1)$ of G. Consequently G has a DHC iff G' has a DHC, and the proof is completed. (The straightforward 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

$$\text{RUHC}=\{e(G); e(P): P \text{ is a UHP in } G \text{ and } G \text{ has an UHC}\}.$$

THEOREM 3. RUHC *is* NP-*complete.*

Proof. We reduce the RDHC to it. The construction is identical to the one used in the proof of the NP-completeness of the ordinary UHC problem [1], [4]. 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

$$M=\left\{ e(c); e(i, j): \begin{array}{l}\text{the edge } (i, j) \text{ does not appear in any} \\ \text{optimal tour of the instance } c \text{ of the TSP}\end{array} \right\}.$$

THEOREM 4. *M is* NP-*complete.*

Proof. We reduce the RUHC to it. Let $(G; P)$ be an instance of the RUHC, where $P=(w_1, w_2, \cdots, w_n)$ is a 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 $(G, P) \in \text{RUHC}$, then G has a UHC and hence (w_1, w_n) (which, by definition of a 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 a UHC. Consequently $(c, (i, j)) \in M$ iff $(G, P) \in$ RUHC. □

We now define the following language:

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

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 could require iterations of complexity more than polynomial:

THEOREM 5. *L_0 is* NP-*complete.*

Proof. We reduce 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* [9] if $(c(f) - c(s))/c(s) \leqq \varepsilon$. Otherwise f is called *ε-suboptimal.* In a similar way to L_0, the following language is defined for $\varepsilon > 0$:

$$L_\varepsilon = \{e(c); e(f) : f \text{ is an } \varepsilon\text{-suboptimal tour for the instance } c\}.$$

THEOREM 6. *L_ε is* NP-*complete for all $\varepsilon > 0$.*

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 $c(v, u) = 2 + |V|\varepsilon$ otherwise. f is again the tour corresponding to P. It can be easily seen that $(G, P) \in$ RUHC iff $(c, f) \in L_\varepsilon$. □

We say that a local search algorithm is *ε-approximate* if all local optima produced by this algorithm are ε-approximate. The following theorem suggests that local search algorithms for the TSP with iterations requiring only a polynomial amount of time (such as the ones proposed by [5], [6], [11], [12]) will yield local optima of no guaranteed accuracy.

THEOREM 7. *If $P \neq NP$, local search algorithms having polynomial complexity per iteration cannot be ε-approximate for any $\varepsilon > 0$.*

Proof. It suffices to show how, by using the function IMPROVE of an ε-approximate local search algorithm, we can solve the RUHC problem. Given an instance $(G = (V, E), P)$ of this problem, we construct, as before, the instance c of the $|V|$-city TSP with $c(u, v) = 1$ if $(u, v) \in E$, and $c(u, v) = 2 + |V|\varepsilon$ otherwise, and f, the tour corresponding to the Hamiltonian path P. f has cost $|V|(1 + \varepsilon) + 1$; moreover there is no tour of better cost, unless G has a UHC. Hence $(G, P) \in$ RUHC iff IMPROVE $(f, c) \neq$ 'no'. □

It should be emphasized that Theorem 7 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, there are polynomial-time algorithms (not necessarily of iterative nature) yielding 1-approximate [8] and $\frac{1}{2}$-approximate [3] solutions.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.

[2] M. BLUM, R. W. FLOYD, V. R. PRATT, R. L. RIVEST AND R. E. TARJAN, *Time bounds for selection*, J. Comput. System Sci., 7 (1972), pp.448–461.

[3] N. CHRISTOFIDES, Private communication, March 1976.

[4] R. M. KARP, *Reducibility among combinatorial problems*, Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, pp. 85–103.

[5] S. LIN, *Computer solutions of the traveling salesman problem*, Bell System Tech. J., 44 (1965), pp. 2245–2269.

[6] S. LIN AND B. W. KERNIGHAN, *An effective heuristic algorithm for the traveling salesman problem*, Operations Res., 21 (1973), pp. 498–516.

[7] C. H. PAPADIMITRIOU, *The complexity of the local structure of certain convex polytopes*, Proc. 1976 Conf. on Information Systems and Sci., Johns Hopkins Univ., Baltimore, Md., March 31–April 2, 1976, pp. 47–51.

[8] D. J. ROSENKRANTZ, R. E. STEARNS AND P. M. LEWIS, *Approximate algorithms for the traveling salesperson problem*, IEEE 15th Annual Symp. on Switching and Automata Theory, Univ. of New Orleans, New Orleans, La., Oct. 14–16, 1974, pp. 33–42.

[9] S. SAHNI AND T. GONZALES, *P-complete approximation problems*, J. Assoc. Comput. Mach., 23 (1976), pp. 555–565.

[10] S. L. SAVAGE, P. WEINER AND M. J. KRONE, *Convergent local search*, RR #14, Dept. of Comput. Sci., Yale Univ., New Haven, Conn., 1973.

[11] S. REITER AND G. S. SHERMAN, *Discrete optimizing*, SIAM J. Appl. Math., 13 (1965), pp. 864–889.

[12] K. STEIGLITZ AND P. WEINER, *Some improved algorithms for computer solution of the traveling salesman problem*, Proc. 6th Ann. Allerton Conf. on Circuit and System Theory, Univ. of Ill., Urbana, Ill., Oct. 1968, pp. 814–821.

[13] P. WEINER, S. L. SAVAGE AND A. BAGGHI, *Neighborhood search algorithms for guaranteeing optimal traveling salesman tours must be inefficient*, J. Comput. System Sci., 12 (1976), pp. 25–35.