Neighborhood Search Algorithms for Finding Optimal Traveling Salesman Tours Must Be Inefficient

P. Weiner, S. L. Savage, A. Bagchi

Research Report #13

This paper will be presented at the Fifth Annual ACM Symposium on Theory of Computing, to be held in Austin, Texas, April 1973

March 1973

Neighborhood Search Algorithms for Finding Optimal Traveling Salesman Tours Must Be Inefficient

P. Weiner, S. L. Savage, A. Bagchi*
Department of Computer Science, Yale University†

I. Introduction

In this paper, we explore the use of *neighborhood search techniques* for finding *optimal* solutions to the symmetric Traveling Salesman Problem. These techniques have been dramatically successful in obtaining *near-optimal* solutions to this problem for a reasonable expenditure of effort (1,2,3,4,5,6,9,10,12). Extensions of these techniques can be used to obtain the globally optimum solution, but the effort involved is at least an exponential function of the number of cities, n. Indeed, as this paper demonstrates, all local search algorithms that are capable of finding the optimal solution to an arbitrary n-city problem must grow at least as fast as $\left(\frac{n-2}{2}\right)!$. Thus for large problems, these algorithms are computationally inefficient.

In the following sections we show that any exact neighborhood search algorithm for the Traveling Salesman Problem must inspect a prohibitively large number of feasible solutions. We begin with a brief discussion of the Traveling Salesman Problem (TSP) and neighborhood search techniques in section II. In section III we develop a necessary condition for neighborhood search to converge to an optimal solution. We use this result in sections IV and V to obtain a lower bound on the effectiveness of neighborhood search as applied to the TSP.

II. The TSP and Heuristic Search Algorithms

The TSP can be described briefly as follows. Given a set of n cities, consider the weighted graph formed by taking the cities as nodes, with the arc length between nodes i and j defined to be the distance between cities i and j. The *feasible solutions* are represented by the $\frac{(n-1)!}{2}$ Hamiltonian circuits. The *cost* of a feasible solution is the sum of the lengths of its arcs, or inter-city links. Given the $\binom{n}{2}$ arc lengths, the *optimal feasible solution* is the Hamiltonian circuit of minimum cost. We refer to a particular assignment of arc lengths as the *parameter* x of the problem, and assume for simplicity that x can take on any set of $\binom{n}{2}$ real values. Although the use of some other reasonable parameter set might cause serious technical difficulties, it would not strongly affect the conclusions reached below. It is easy to show, for example, that restricting the $\binom{n}{2}$ values of x to be positive is of no consequence i the following theory. We write c(s,x) for the cost of the feasible solution s with respect to the parameter x. For a more detailed discussion of the TSP we recommend (13).

As with any combinatorial problem, it is of interest to determine the growth rate of proposed algorithms as a function of a key variable, here taken to be the number of cities, n.

To date, there is no known algorithm for solving the TSP that grows less than exponentially with the number of cities. It is not surprising that a number of fast-running heuristic procedures have been developed that produce solutions that may not be optimal. We refer to those algorithms that always find the optimal solution as *exact algorithms*, and to the heuristic procedures as *approximate algorithms*. A particularly effective class of approximate algorithms may be described as *neighborhood search algorithms*.

We now give an informal definition of neighborhood search. More discussion can be found in (11), (15), and (17). Let S_n denote the set

* Current address of A. Bagchi: 135a Bangor Avenue Calcutta 35, India

† This work was partially supported by a grant from the Mobil Foundation to P. Weiner and by an IBM Postdoctoral Fellowship awarded to A. Bagchi.

feasible solutions associated with the n-city problem. For every solution $s \in S_n$, a subset or *neighborhood* of S_n, $N(s)$, is defined. When such a neighborhood has been defined for each $s \in S_n$, we say that a *neighborhood structure* N has been defined on S_n. Given a specific parameter, x, a sequence of solutions in S_n is then generated as follows.

s_1, the initial solution, is arbitrary.

s_{i+1} can be any point in $N(s_i)$ such that $c(s_{i+1},x) < c(s_i,x)$.

When for some k, $c(s_k,x) \leq c(s,x)$ for all $s \in N(s_k)$, s_k is said to be *locally optimal* with respect to the structure N. Note that s_k is not necessarily globally optimal, but the cost of elements of the sequence is strictly decreasing.

We have not discussed the procedure by which s_1 is chosen, or the order in which the solutions in $N(s_i)$ are searched for the improvement s_{i+1}. In practice, these choices are usually pseudo-random. Indeed, the algorithm may be repeated on many different random starts, producing in general several different local optima of which the best is chosen as the final solution.

A local search algorithm developed by Lin (12) in 1965 is one of the best computational methods for the TSP known today. At roughly the same time, Reiter and Sherman (15) also had some success with similar algorithms and formalized the concept of neighborhood search.

It is important to note that with all neighborhood search algorithms the neighborhood structure is fixed prior to presenting a given instance of the problem (that is, the parameter) to the algorithm. Particular algorithms will differ in the method used to select s_1 and in the strategy used to search $N(s_i)$ for any given solution s_i.

In comparing different algorithms, we choose to count the worst-case number of solutions examined as the complexity measure. While the inherent pseudo-random nature of neighborhood search algorithms makes this measure difficult to determine precisely, we can bound it with the following observation. Suppose that our algorithm is fortunate enough always to pick the optimal solution as s_1 (a friendly demon is at work!). We would still have to explore all of $N(s_1)$ before terminating to ascertain that s_1 was indeed optimal. The symmetry of the parameter set implies that all feasible solutions have a chance of being optimal, so the worst-case behavior is bounded from below by

$$\max_{\text{feasible } s_i} |N(s_i)|.*$$

We use this fact to bound the complexity of exact local search algorithms in section V.

Definition: A neighborhood structure N is *exact* if for any parameter x and any $s \in S_n$,

$$c(s_i,x) \leq c(s,x) \text{ for all } s \in N(s_i)$$

$$\Rightarrow s_i \text{ is optimal.}$$

In other words, a neighborhood structure N is exact provided any local optimum with respect to N is a global optimum. It is easy to see that if $N(s_i) = S_n$ for all s_i, the corresponding algorithm is in fact exact, but that the complexity of this algorithm is bounded from below by $\frac{(n-1)!}{2}$ for the TSP. For smaller neighborhood structures, the search algorithm may not be exact, and it becomes possible to produce locally optimum solutions. Our desire to consider those neighborhood structures that are just sufficient to correspond to exact algorithms motivates the next section.

III. A Theorem Concerning Exactness

We now introduce a result relating to the exactness of neighborhood structures (16,17). Although this result actually applies to a wide class of problems of which the TSP is but one example, we phrase it here in terms of the TSP. See (16) for more general statements of both Theorem 1 and Theorem 2 below.

Definition: Given a feasible solution s_j, we define as $O(s_j)$ that set of solutions $s_k \in S_n$ for which there exists an x such that

$$c(s_k,x) < c(s_j,x) \leq c(s_i,x) \text{ for all } i \neq j,k.$$

In other words, $O(s_j)$ consists of those solutions that can be uniquely optimal when s_j is second to optimal.

The feasible solutions of the TSP are uniquely determined by the order in which the cities are arrived at in the Hamiltonian circuit. By merely renaming the cities of the problem, we can transform any feasible solution to any other feasible solution. This symmetry implies that the size of O(s) does not vary with s.

Theorem 1: The minimal exact neighborhood

* We use $|P|$ to denote the cardinality of the set P.

structure for the TSP is unique and consists of $O(s)$ for each $s \in S_n$.

Proof: We first show that if N is exact, then $O(s) \subseteq N(s)$ for every $s \in S$. We prove the contrapositive. For some s, assume that there is an $s' \in O(s)$ for which $s' \notin N(s)$. Then by the definition of $O(s)$ there exists a parameter x such that s' is uniquely optimal and s is second to optimal. Now suppose for this x, s happens to be chosen as the initial feasible solution. Then s will be locally optimal with respect to N, but it is not globally optimal, a contradiction.

We have thus shown that if the neighborhood structure N is exact, $N(s)$ must contain $O(s)$ for each s. We now show that the neighborhood structure comprising $O(s)$ for each s is in fact exact. This is equivalent to showing that if s is non-optimal then some element of $O(s)$ has lower cost than s. Assume therefore that there exists a parameter x_1 for which s is non-optimal and for which all solutions in $O(s)$ have higher cost. For the sake of contradiction, we construct another parameter for which some $s' \notin O(s)$ is uniquely optimal with s second to optimal. Throughout the construction we denote the length of the k^{th} inter-city link by $x[k]$, which gives

$$c(s,x) = \sum_{k \in s} x[k].$$

We define as $L_s(x)$ the set $\{s_i | c(s_i x) < c(s,x)\}$, and assume for the moment that the solutions in $L_s(x)$ are not all tied in cost.

We now construct a finite sequence of x_i's such that $|L_s(x_i)|$ monotonically decreases until for some x_j, $|L_s(x_j)| = 1$. This implies that the one solution s' remaining in $L_s(x_j)$ belongs to $O(s)$, contradicting the hypothesis.

Start with the parameter x_1 above, and proceed iteratively as follows. At the i^{th} stage we let s_i denote some solution with maximum cost within $L_s(x_i)$. (By assumption if there are no solutions in $L_s(x_i)$ with cost less than s_i, then s_i is the only solution left, and we are done.) Since the feasible solutions all consist of exactly n links, every solution in $L_s(x_i)$ contains some link not contained in s. Let k denote such a link contained in s_i but not s. We can raise $x[k]$ without raising $c(s,x)$ until $c(s_i,x) > c(s,x)$. This new parameter is x_{i+1}. Note that in raising $x[k]$, we can only raise the cost of other solutions with respect to s, insuring that $L_s(x_{i+1})$ is a non-empty proper subset of $L_s(x_i)$, so that

$$1 \leq |L_s(x_{i+1})| < |L_s(x_i)|.$$

Since $L_s(x_1)$ was finite, we must eventually arrive at a parameter x_j, such that $|L_s(x_j)| = 1$.

We now justify the assumption concerning ties in L_s. If for any x such a tie should occur, we can break it in the following manner. Let h denote $c(s,x) - c(s_i,x)$, where s_i is any solution in $L_s(x)$. Every feasible solution consists of exactly n links. So given any pair of solutions s_i and s_j in $L_s(x)$, there exists a link $k \in s_i$ such that $k \notin s_j$. If $k \notin s$ we can break the tie while preserving the set $L_s(x)$, by adding $\frac{h}{2}$ to $x[k]$. On the other hand if $k \in s$ we can accomplish this result by subtracting $\frac{h}{2}$ from $x[k]$.

QED

IV. Primary Changes

We have been unable to characterize explicitly the complete O-neighborhoods for the TSP. In this section we provide some important definitions and prove some preliminary results concerning certain feasible solutions that must be contained in $O(s)$. These results allow us in section V to calculate a lower bound on the size of the complete neighborhoods, and hence on the complexity of the exact algorithms.

Definition: If s and s' are two TSP solutions such that s' can be produced from s by exchanging k links in s with k links not in s, we say that s and s' are k-*changes* of one another.

Definition: Let s' be a k-change of s where $A = \{a_1,\dots,a_k\}$ is the set of inter-city links belonging to s but not to s'. If no two elements of A are *adjacent*, that is, incident to the same city, we say that the set A is *nonadjacent*, and call s' a *nonadjacent* k-*change* of s.

Given a feasible TSP solution s, let $A = \{a_1,\dots,a_k\}$ be a set of links belonging to s, and $B = \{b_1,\dots,b_k\}$ be a set of links not belonging to s. We denote by $G(A,B)$ the graph whose vertex set corresponds to the set of links A, with an edge connecting node i and node j iff some $b \in B$ is adjacent to both link a_i and link a_j. We write $s - A + B$ to mean the set of inter-city links obtained by the removal of the set A and the addition of the set B to the solution s.

Lemma 1: If $s' = s - A + B$ is a nonadjacent k-change of s, then $G(A,B)$ consists of k edges forming one or more disjoint cycles.

Proof: Assume that s' is a nonadjacent change of s, and assume that some $b \in B$ is the link between cities m and n. In order that these cities end up with exactly two incident links both m and n must have had exactly one link removed. Thus we have each end of b adjacent to a link of A which implies that each of the k b's appears in G. Now assume that some element $a \in A$ links cities i and j. Since we assume that no other element in A is incident to either i or j, and since in a feasible solution every city must have exactly two incident links, B must contain exactly one link incident to i and one to j. Thus each end of the link a is adjacent to exactly one link in B. This implies that all the vertices of G have a degree (the number of incident edges) of two, which in turn implies that G consists of disjoint cycles.

QED

Definition: If s' is a nonadjacent change of s such that $G(A,B)$ consists of a single cycle then s' is a *primary* change of s.

Theorem 2: Let s and s' be feasible solutions to the n-city problem. If s' is a *primary* k-change of s then $s' \in O(s)$.

Proof: We construct a parameter for which s' is uniquely optimal with s uniquely second in cost to s'. By hypothesis $s' = s - A + B$, where

$A = \{a_1,\ldots,a_k\}$, $B = \{b_1,\ldots,b_k\}$, and $A \cap B = \Phi$.

Assign lengths to the links as follows. Letting ε be a number such that

$$0 < \varepsilon < \frac{1}{2k-1},$$

set

$$x[i] = \begin{array}{l} 0, \text{ for } i \in s' \cap s \\ 1-\varepsilon, \text{ for } i \in B \\ 1+\varepsilon, \text{ for } i \in A \\ 2, \text{ otherwise.} \end{array}$$

This gives

(1) $c(s',x) = k\cdot(1-\varepsilon)$

and

(2) $c(s,x) = k\cdot(1+\varepsilon) = c(s',x) + 2k\varepsilon$.

Clearly s' is uniquely optimal because it contains the n smallest links, k of cost $1-\varepsilon$ and n-k of cost 0. We now show that s is uniquely second in cost to s'. We begin by showing that if a solution contains any links outside of $s \cup s'$, or does not contain all links in $s \cap s'$, then it must have a cost greater than $c(s,x)$.

Let s_1 be a solution containing a link outside of $s \cup s' = A \cup s'$. The lowest cost solution of this form would use all the 0 cost links in s' and would replace one link of cost $1-\varepsilon$ by a link of cost 2. This would give

$$\begin{aligned} c(s_1,x) &\geq c(s',x) - (1-\varepsilon) + 2 \\ &= c(s',x) + 1 + \varepsilon. \end{aligned}$$

But we have chosen ε so that $2k\varepsilon < 1+\varepsilon$, so by (2) the cost of s_1 is greater than both s' and s.

Let s_2 be a solution not containing some link in $s \cap s'$. The lowest cost solution of this form must replace some link in s' of cost 0 with a link of cost $1+\varepsilon$, yielding

$$\begin{aligned} c(s_2,x) &\geq c(s',x) + (1+\varepsilon) - 0 > c(s',x) + 2k\varepsilon \\ &= c(s,x). \end{aligned}$$

So s_2 has a cost greater than both s and s'.

Thus for a solution to lie between s and s' in cost, it must be of the form $s_1 = s - A' + B'$, where $A' \subseteq A$ and $B' \subseteq B$. Hence by Lemma 1 B' forms a cycle in $G(A,B)$. But by hypothesis B forms the only cycle in $G(A,B)$ because s' is a primary change of s. We may conclude that the preceeding assignment of link costs renders s' uniquely optimal with s uniquely second in cost to s', implying that $s' \in O(s)$.

QED

V. A Lower Bound on the Size of O(s)

We now find a lower bound on the number of primary changes of a TSP solution, and hence a lower bound on O(s).

Given a solution s, a nonadjacent set $A = \{a_1,\ldots,a_k\}$ belonging to s, and any cycle on the set A, a primary change of s of the form $s - A + B$ can be constructed such that $G(A,B)$ consists of that cycle (see the appendix for details). This gives us a distinct solution in O(s) for each distinct cycle on $G(A,B)$.

Let K(n) denote the size of the largest nonadjacent set that can be removed from an n-city solution, and M(n) denote the number of such nonadjacent sets. Since the number of distinct cycles on k points is $\frac{(k-1)!}{2}$, a lower bound on the number of primary changes for an n-city problem is

$$M(n) \cdot \frac{(K(n)-1)!}{2}.$$

It is apparent from figures 1.a and 1.b that K(n) is $\frac{n}{2}$ or $\frac{n-1}{2}$ for even or odd n respectively. It is also clear that $M(n) = 2$ for even n. For odd n notice that the two adjacent links (1 and 2 in figure 1.b) uniquely determine the $\frac{n-1}{2}$ links to be removed. The possible number of such adjacent links is n, so that when n is odd, $M(n) = n$. Therefore a lower bound on the size of O(s) is

$$2 \cdot \frac{(n/2 - 1)!}{2} = \left(\frac{n-2}{2}\right)!, \text{ n even,}$$

and

$$n \cdot \frac{((n-1)/2 - 1)!}{2} = \frac{n}{n-1} \cdot \left(\frac{n-1}{2}\right)!, \text{ n odd.}$$

With this result we have established that the time required to search only the last neighborhood arrived at in an exact algorithm, and thereby guarantee optimality, is proportional to at least

$\left(\frac{n-2}{2}\right)!$ rendering exact neighborhood search

impractical for this problem.

Acknowledgement

We wish to acknowledge the contribution of Martin J. Krone, who initiated our study of exact neighborhood structures and also provided numerous details concerning the Traveling Salesman Problem.

References

1. F. Bock, An Algorithm for Solving "Traveling-Salesman" and Related Network Optimization Problems, presented at the 14th National Meeting of ORSA, 1958.
2. M. Bellmore and G. Nemhauser, The Traveling Salesman Problem: A Survey, Operations Research 16 (3), 1968, 538-558.
3. N. Burkov and S. Lovetskii, Methods for the Solution of Extremal Problems of Combinatorial Type, Aut Remot RR, 1968, 1785-1806.
4. G. Croes, A Method for Solving Traveling-Salesman Problems, Operations Research 6 (6), 1958, 792-812.
5. G. Dantzig, R. Fulkerson, and S. Johnson, Solution of a Large-Scale Traveling-Salesman Problem, Operations Research 2 (4), 1954, 393-410.
6. M. Flood, The Traveling-Salesman Problem, Operations Research 4 (1), 1956, 61-75.
7. W. D. Frazer, Analysis of Combinatory Algorithms -- A Sample of Current Methodology, AFIPS Conference Proceedings 40, 1972, 483-491.
8. R. L. Graham, Bounds on Multiprocessing Anomalies and Related Packing Algorithms, AFIPS Conference Proceedings 40, 1972, 205-217.
9. M. Held and R. Karp, A Dynamic Programming Approach to Sequencing Problems, SIAM Journal 10, 1962, 196-209.
10. R. Karg and G. Thompson, A Heuristic Approach to Solving Traveling Salesman Problems, Management Science 10, 1964, 225-248.
11. M. Krone, Heuristic Programming Applied to Scheduling Problems, Princeton University PhD Thesis, September 1970.
12. S. Lin, Computer Solutions of the Traveling Salesman Problem, Bell System Tech Journal 44, 1965, 2245-2269.
13. C. L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill, 1968, 181.
14. E. M. Reingold, Establishing Lower Bounds on Algorithms: A Survey, AFIPS Conference Proceedings 40, 1972, 471-481.
15. S. Reiter and G. Sherman, Discrete Optimizing, J. Soc. Indust. Appl. Math. 13 (3), 1965, 864-889.
16. S. L. Savage, The Solution of Discrete Linear Optimization Problems by Neighborhood Search Techniques, Doctoral Dissertation, Department of Computer Science, Yale University, 1973.
17. S. L. Savage, P. Weiner, M. J. Krone, Towards a Theory of Convergent Local Search, 6th Princeton Conference on Information Science and Systems, 1972.
18. J. D. Ullman, The Performance of a Memory Allocation Algorithm, Tech Report 100, Electrical Engineering Department, Princeton University, 1971.

Appendix: The Construction of Primary Changes

A cycle on a set of vertices can be specified by the order in which the vertices are arrived at in a traversal of the circuit. Given a feasible TSP solution s, a nonadjacent set of links $A = \{a(1),\ldots,a(k)\}$, and a permutation $\pi_1,\ldots,\pi_k$,

we now describe a procedure for constructing a solution s − A + B for which the order of the vertices around the circuit G(A,B) is

$a(\pi_1), a(\pi_2),\ldots,a(\pi_k)$.

We proceed as follows.

(1) Remove $a(\pi_1)$ from s.

(2) Denote one end of $a(\pi_1)$ by H_1 and the other

by T_1. Set i to 2.

(3) Starting at T_1 traverse the links presently

in the path until $a(\pi_1)$ is reached. Denote the

end of $a(\pi_1)$ reached first by T_i and the other b

H_i. Remove $a(\pi_i)$. Let b_{i-1} be the link from H_i

to T_i. If i = k go to (4). Otherwise go to (3)

(4) b_k is the link from H_k to T_1.

In figure 2 the steps above are depicted usin elastic links. Figure 3 shows the complete construction of a primary 4-change by the method above.

It is apparent from the construction that the circuit G(A,B) can be traversed so as to arrive the vertices a_i in the same order as the links a

were removed in the procedure above. Since this order was arbitrary to begin with, in this manne we can produce a primary solution s − A + B for which the vertices of G(A,B) lie in any of the

$\frac{(k-1)!}{2}$ possible orderings.

[illegible] totally nonadjacent sets of removed links.

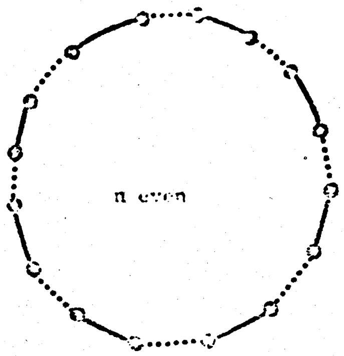


Fig. 1a

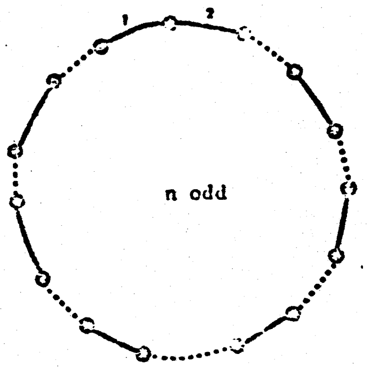
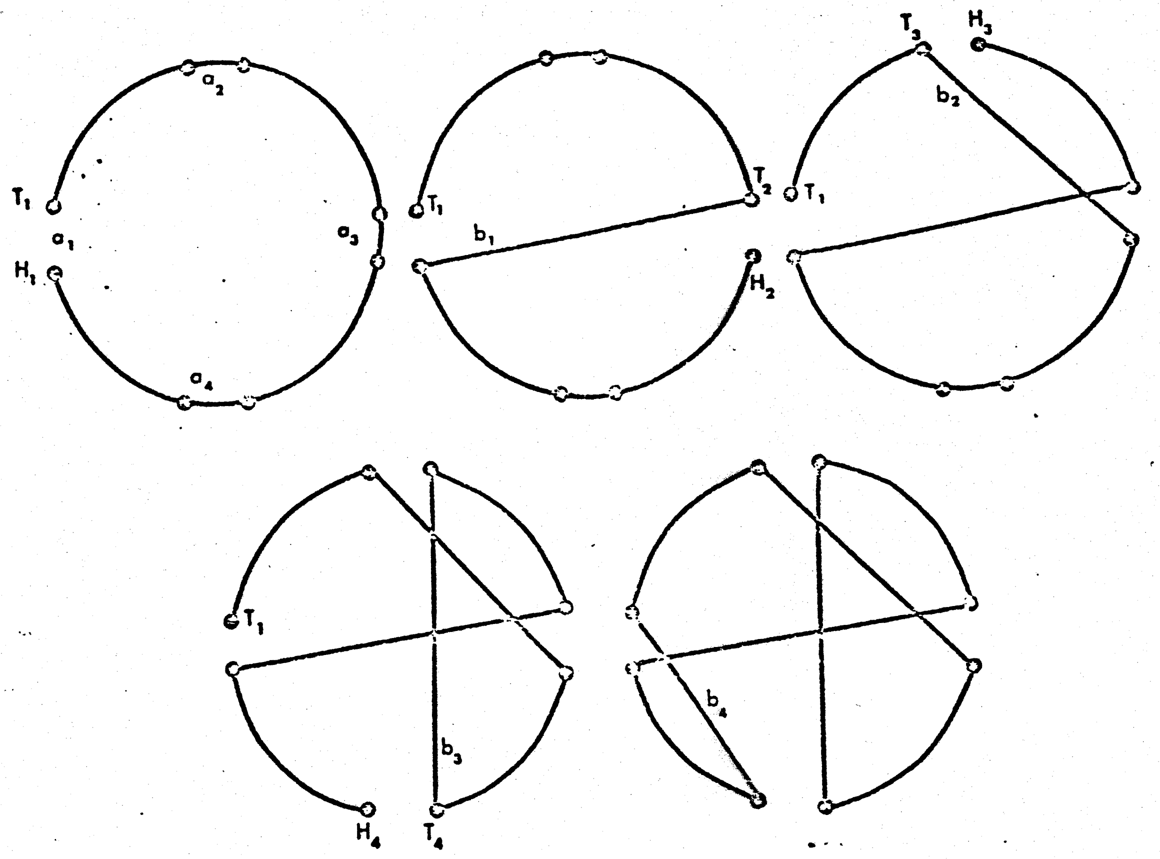


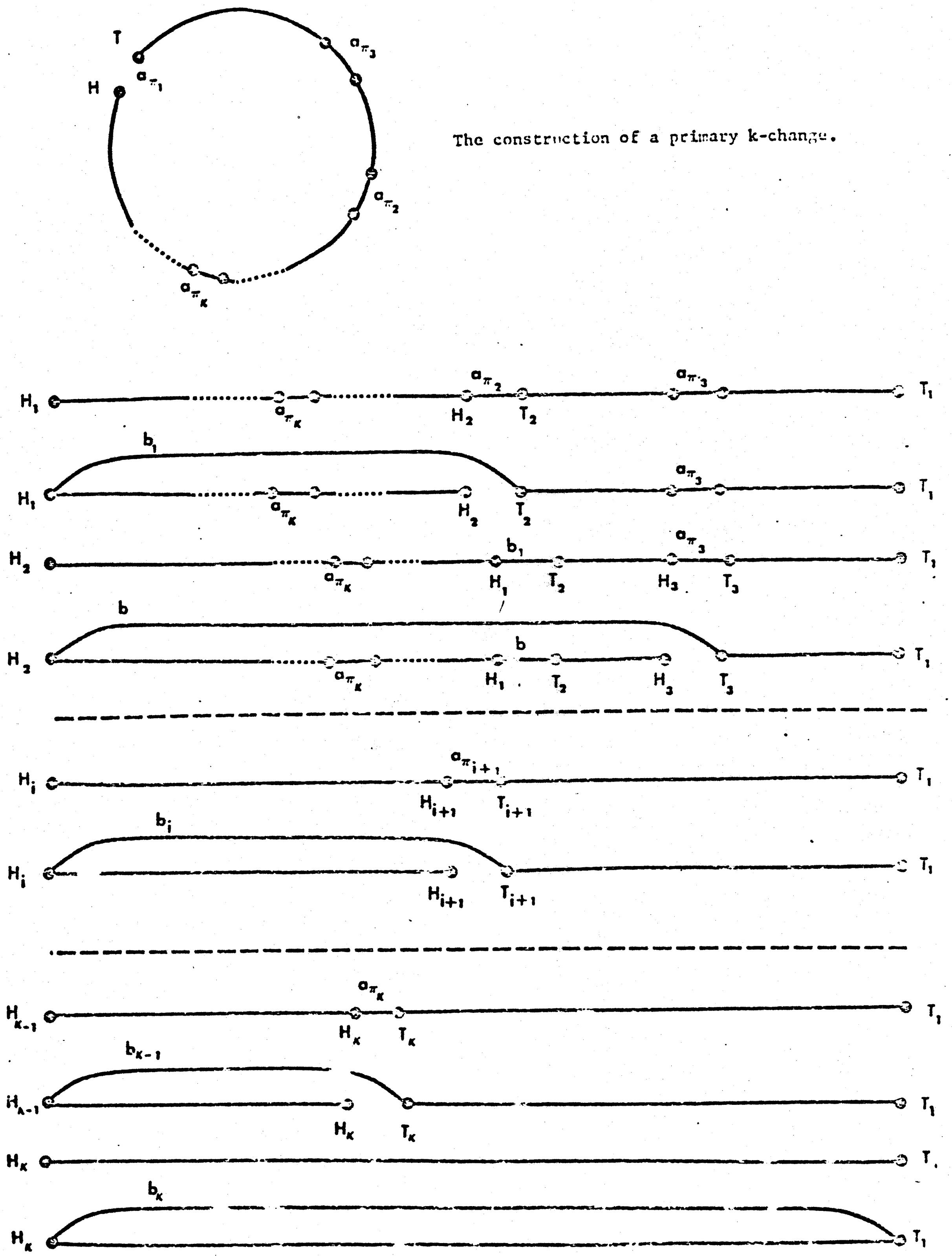
Fig. 1b

Fig. 2 is on the following page.



The construction of a primary 4-change with π = 1 3 2 4.

Fig. 3



The construction of a primary k-change.

Fig. 2