AN ALGORITHM FOR THE MANY-VISITS m-TRAVELING SALESMAN PROBLEM

Jayme Luiz Szwarcfiter

NCE 0285

November 1985

Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
Brasil

AN ALGORITHM FOR THE MANY-VISITS m-TRAVELING SALESMAN PROBLEM

Jayme Luiz Szwarcfiter
Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica

ABSTRACT

We described a method based on integer linear programming for solving the many-visits m-traveling salesman problem.

RESUMO

Descreve-se um método baseado em programação linear inteira para a solução do problema do m-caixeiro viajante com múltiplas visitas.

1. INTRODUCTION

We describe an integer linear programming (ILP) approach to the many-visits traveling salesman problem of [2, 5]. The method is based on [3] and is an application of [4]. As usual with ILP formulations of the TSP a central point is handling subtours. The constraints described in this paper do not eliminate them necessarily and instead allow the existence of certain subtours, which are subsequently transformed into proper tours of the same cost.

The m-TSP arises naturally when considering many-visits type problems. That is, there is a distinguished city which is to be visited m times, once by each salesman. In addition, it is useful, for instance, when solving the aircraft scheduling problem (ASP) with m runways. This problem is to find a landing schedule of a set of n airplanes so as to minimize the time of the latest landing. Each plane is of a certain class and the time gap required between two consecutive landings on a same runway depends only on the two classes involved. The number k of distinct classes is supposed to be fixed [2, 5]. In [5], the $m > 1$ case has been discussed, but only solutions for $m=2$ have been given. The ASP with m runways can be solved by an algorithm for the many-visits m-TSP.

For convenience, we adopt the interpretation considering the many-visits m-TSP as a special case of the ordinary m-TSP That is, we formulate an algorithm for solving a general asymmetric m-TSP which starts by dividing the n vertices into k disjoint classes C_i, according to a certain equivalence relation. In the many-visits m-TSP, each C_i would correspond to a city to be visited $|C_i|$ times. If k is fixed and the maximum distance of the problem is bounded by a polynomial in n, the algorithm terminates within linear time in the size of its input and output.

2. THE ALGORITHM

D is a digraph having vertices $V(D) = \{v_1,\dots,v_n\}$, $n>1$ and v_n a distinguished vertex. There is an edge $(v_p,v_q) \in E(D)$ with a non negative integer d_{pq} associated to it, for every pair of distinct $v_p,v_q \in V(D)$. The value d_{pq} is the distance from v_p to v_q. The cost of a subdigraph D' of D is the sum $\Sigma\, d_{pq}$, for all $(v_p,v_q) \in E(D')$.

Let m be an integer, $1 \le m \le n-1$. A tour is a minimum cost subdigraph T of D such that

(i) T is a union of m cycles of D

and

(ii) v_n belongs to all cycles of T, but if $i<n$ then v_i is in exactly one cycle.

The m-TSP consists of finding T, given D and m.

Let D^+ be the digraph obtained from D by adding to it m-1 vertices $v_{n+1},\dots,v_{n+m-1}$ and edges from (to) each v_i, $n+1 \le i \le n+m-1$, to (from) every other vertex of D^+, with distances satisfying

$d_{pq} = \infty$, for $n \le p,q \le n+m-1$ and $p \ne q$

$d_{pq} = d_{nq}$ and $d_{qp} = d_{qn}$, for $p \ge n$ and $q<n$.

The vertices $v_n,v_{n+1},\dots,v_{n+m-1}$ are the roots of D^+.

Let $C_1 \cup \ldots \cup C_k = V(D^+)$ be a partition of the vertices of D^+ into disjoint non-empty subsets, called <u>classes</u> of D, such that $v_p, v_q \in V(D^+)$ belong to the same class iff

$$d_{pq} = d_{qp} \quad \ldots . (1)$$

$$d_{hp} = d_{hq} \text{ and } d_{ph} = d_{qh}, \text{ for all } h \neq p,q,\ 1 \leq h \leq n+m-1 \quad \ldots . (2)$$

Without loss of generality, assume $C_k = \{v_n, \ldots, v_{n+m-1}\}$.

Let C_i, C_j be two (not necessarily distinct) classes of D and choose $v_p \in C_i$ and $v_q \in C_j$, with $p \neq q$. Define $\tau_{ij} = d_{pq}$. Clearly, τ_{ij} is independent of the particular pair v_p, v_q chosen.

The <u>profile</u> of a subdigraph D' of D^+ is a matrix $F(D')$ in which each element (i,j), $1 \leq i,j \leq k$, equals the number of edges from C_i to C_j in D'.

Given the classes of D, we show that the profile of some tour of D can be computed by the following ILP problem, with variables x_{ij}, $1 \leq i,j \leq k$.

$$\text{minimize} \sum_{1 \leq i,j \leq k} \tau_{ij} x_{ij} \quad \ldots . (3)$$

$$\text{s.t. } x_{ij} \text{ is an integer } \geq 0 \quad \ldots . (4)$$

$$\sum_{1 \leq j \leq k} x_{ij} = \sum_{1 \leq j \leq k} x_{ji} = |C_i|,\ 1 \leq i \leq k \quad \ldots . (5)$$

$$\sum_{i,j \in S} < \sum_{i \in S} |C_i|,$$

$$\text{for all subsets } S \subset \{1, \ldots, k-1\}. \quad \ldots .. (6)$$

Theorem 1: Let $X=(x_{ij})$ satisfy (3)-(6). Then there exists a tour T such that $F(T)=X$. In addition, T can be computed in $O(nk)$ steps, given X.

The proof consists of constructing T.

Let X satisfy (3)-(6). A subtour T^* for X is a spanning subdigraph of D^+ such that

(i) $F(T^*)=X$

and

(ii) T^* is a union of vertex disjoint cycles of D^+.

The following algorithm constructs a subtour for X.

ALGORITHM 1: Constructing T^*

Initial step: Given x_{ij}, $1 \le i,j \le k$, and the partition of $V(D^+)$ into classes $C_1,\ldots,C_k$, define $E(T^*):=\phi$ and for all $u \in V$, $in(u):=out(u):=0$.

General step: If $x_{ij}=0$ for all $1 \le i,j \le k$ the process terminates (T^* is ready). Otherwise choose arbitrarily i,j such that $x_{ij} > 0$. Denote by $t(u)$ the sum $in(u)+out(u)$, $u \in V(D^+)$. Select two distinct vertices $v \in C_i$ and $w \in C_j$ satisfying

$$out(v)=in(w)=0 \quad \ldots. (7)$$

$$in(v)\neq 0 \Rightarrow t(v')\neq 0, \text{ for all } v' \in C_i,\ v'\neq w \quad \ldots. (8)$$

$$out(w)\neq 0 \Rightarrow t(w')\neq 0, \text{ for all } w' \in C_j,\ w'\neq v \quad \ldots. (9)$$

Then define $E(T^*):=E(T^*)\cup\{(v,w)\}$ and $out(v):=in(w):=1$, decrease x_{ij} by 1 and repeat the general step.

Lemma 1: Algorithm 1 constructs a subtour T^* for X.

Proof: First, we show by induction that in each iteration ℓ there is a pair of vertices satisfying (7)-(9). This is clear for $\ell=1$. Suppose $1<\ell\leq n+m-1$. From (5), it follows we can choose $v \in C_i$, $w \in C_j$ such that $out(v)=in(w)=0$, for every ℓ. It remains to prove that there is a choice with $v\neq w$. If $i\neq j$ this is trivial. Suppose $i=j$ and that (7) holds only for $v=w$. Then there exists $u \in C_i$, $u\neq v$ with $t(u)=2$, otherwise $|C_i|=1$, $x_{ij}=1$, contradicting (6). In the iteration that chose u last, $t(u)=1$ and $t(v)=0$, which contradicts (8) or (9), and the induction hypothesis. In addition, if v satisfies 7) and not (8) there is $v' \in C_i$, $v'\neq w$, with $t(v')=0$. In this case, replace v by v'. Similarly for (9) and we conclude that suitable v,w exist for $1\leq\ell\leq n+m-1$. Each iteration decreases some $x_{ij} > 0$ by 1. From (5), $\Sigma x_{ij}=n+m-1$. Using (4), we conclude that the process terminates after n+m-1 iterations of the general step. Each one adds the edge (v,w) to $E(T^*)$. At termination, $in(u)=out(u)=1$, for all u. The lemma follows.

We now proceed to transform T^* into a proper tour T. With this purpose we describe two operations, SPLIT and JOIN, on the cycles of D^+. Let C(v) be the class of D containing vertex v.

Let Z be the cycle $r_1,\ldots,r_p,s_1,\ldots,s_q,r_1$ of D^+, where $p,q>1$ and the non consecutive vertices r_1,s_1 are such that $C(r_1) = C(s_1)$. Then SPLIT $(Z,r_1,s_1$ removes from Z the edges (r_p,s_1) and (s_q,r_1), while adding to it (r_p,r_1) and (s_q,s_1). The effect of this operation is therefore of split Z into the cycles $Z'=r_1,\ldots,r_p,r_1$ and $Z''=s_1,\ldots,s_q,s_1$.

Conversely, let Z' and Z" be disjoint cycles $r_1,\ldots,r_p,r_1$ and $s_1,\ldots,s_q,s_1$ of D^+, respectively such that $C(r_1)=C(s_1)$.

Then JOIN(Z',Z'',r_1,s_1) removes from Z' and Z" respectively the edges (r_p, r_1) and (s_q, s_1), but adds (r_p,s_1) and (s_q,r_1). That is, Z' and Z" are transformed into the single cycle

$r_1,\ldots,r_p,s_1,\ldots,s_q,r_1$.

Lemma 2: For either SPLIT (Z,r_1,s_1) or JOIN (Z',Z'',r_1,s_1) $cost(Z) = cost(Z') + cost(Z'')$.

Proof: Since $C(r_1)=C(s_1)$, it follows that $d_{r_p r_1}=d_{r_p s_1}$ and $d_{s_q r_1}=d_{s_q s_1}$.

Algorithm 2 transforms the subtour T* into the tour T.

ALGORITHM 2: Constructing T

Initial step: Let T* be a subtour.

General step: If each cycle of T* has exactly one root the process terminates (T=T*). Otherwise, if some cycle Z has two distinct roots v,w then SPLIT(Z,v,w) and repeat the general step. In the remaining case, when a cycle Z' of T* has no root then choose a second cycle Z" such that we can identify vertices $v' \in V(Z')$ and $v'' \in V(Z'')$ satisfying $C(v')=C(v'')$. Then JOIN (Z',Z'',v',v'') and repeat the general step.

Lemma 3: Algorithm 2 constructs a tour T of D.

Proof: Each iteration of the algorithm performs either a SPLIT or JOIN. First we show that the conditions for applying these operations are satisfied. Since the distances among roots are all infinite, no two roots can appear in a row in a solution of (3)-(6). Then the roots v and w of the algorithm are not consecutive in the cycle Z. In addition, $C(v)=C(w)=C_k$. That is, SPLIT (Z,v,w) is correct. Now, suppose there is a cycle Z' having no root and such that there is no vertex outside Z' belonging to the same class as any of Z'. Let $C_{q_1},\ldots,C_{q_\ell}$ be the classes of D represented in Z'. Then $S = \{q_1,\ldots,q_\ell\} \subset \{1,\ldots,k-1\}$ is such that $\sum_{i,j\in S} x_{ij} = \sum_{i\in S} |C_i|$, contradicting (6). Then no such Z' can exist and JOIN is correct. Hence the algorithm constructs a digraph T which is obtained from T* by a sequence of $m-|T^*|+p$ SPLIT followed by p JOIN operations, where p is the number of cycles of T* having no root. That is, T is formed by vertex disjoint cycles, each of them containing exactly one root and spanning all vertices of D^+. In addition, by lemma 2 cost(T)=cost(T*). Therefore, T is a tour.

Algorithms 1 and 2 can be implemented in $O(n)$ and $O(n^2$ time, respectively. The time required by the JOIN operations of the latter dominates its complexity. However, the method can be improved by adopting a systematic way of selecting the pair of cycles to be merged by JOIN, as below. We assume that all required SPLIT operations have already been performed.

In the <u>initial step</u>, construct a bipartite graph G having as vertices the cycles Z_j of T* and the classes C_i of D, denoted $G(Z_j)$ and $G(C_i)$, respectively. G has an edge $(G(Z_j), G(C_i))$ when $V(Z_j)\cap C_i \neq \Phi$ and there is a label $v \in V(Z_j)\cap C_i$ attached to it. G has no other edges. Start by examining each $G(Z_j)$, mark it if $G(Z_j)$ is adjacent to C_k and unmark it otherwise. Next remove all vertices $G(C_i), i\neq k$, having degree one in G. Because of (6), the degree of each $G(Z_j)$ remains ≥ 1. In the <u>general step</u>, terminate the algorithm if all vertices $G(Z_j)$ are marked. Otherwise choose some unmarked $G(Z_p)$ and let $G(C_i)$ and $G(Z_q)$ be two vertices of G at distances one and two from $G(Z_p)$, respectively. Denote by v_p and v_q

the labels of $(G(Z_p), G(C_i))$ and $(G(Z_q), G(C_i))$, respectively. Then JOIN (Z_p, Z_q, v_p, v_q). Identify $G(Z_p)$ and $G(Z_q)$ into a new vertex $G(Z_t)$ in G. Mark $G(Z_t)$ iff $G(Z_q)$ is also marked. Remove parallel edges and all vertices $G(C_i)$, $i \neq k$, whose degree dropped to one in G. Repeat the general step.

The above method performs all required JOIN operations in $O(nk)$ time. A sequence of $O(n)$ SPLIT's can be implemented in $O(n)$ steps. The new version of algorithm 2 runs in $O(nk)$ time.

The m-TSP algorithm is now clear. Given D, m and a bound $L \leq n+m-1$ find the classes $C_1, \ldots, C_k$ and stop if $k > L$. Otherwise formulate (3)-(6) and solve the ILP problem. Then compute algorithm 1 and after 2.

By checking (1)-(2) for every pair of vertices, the classes can be determined in $O(n^3)$ time. Alternatively, the following method is more efficient.

In the <u>initial step</u> define $k:=1$ and label each vertex as open. In the <u>general step</u> stop if $k > L$ (failure) or if there are no open vertices (the partition is ready). Otherwise, choose an arbitrary open vertex v_i, relabel it as closed and define $C_k := \{v_i\}$. Then for each open vertex v_j verify if the pair v_i, v_j satisfies (1)-(2), and if it does include v_j in C_k and relabel v_j as closed. At the end, increase k by 1 and repeat the general step.

The above process finds the classes or reports failure in $O(n^2 L)$ time. Its correctness follows from the fact that $C_1, \ldots, C_k$ are the classes of the equivalence relation defined by (1)-(2).

The ILP problem could be solved by generating all distributions of n+m-1 identical objects into k^2 distinct cells (i,j). The tentative value of each variable x_{ij} equals the number of objects placed in (i,j). For each distribution, check (5)-(6) and at the end select the one minimizing (3). There are $0(n^{k^2-1})$ distributions and $0(2^{k-1}+k^2)$ constraints envolving $0(k^2)$ variables each. Therefore this method would find a solution in $0(k^2 2^k n^{k^2-1})$ time.

However, an ILP problem with R variables and S constraints can be solved by Lenstra's algorithm [4] in $0(2^{R^2}(RS \log a)^{c^R})$ time, where a is the largest coefficient in the problem and c a suitable constant (cf. 1). Hence [4] solves (3)-(6) in $0(2^{k^4}(k^2 2^k \log (n+d_{max}))^{ck^2}$ time, d_{max} the maximum distance.

If k is fixed and $d_{max} = 0(n^b)$ for some constant b, the ILP problem can therefore be solved in less than 0(n) time. In this case, the complexity of the TSP algorithm becomes $0(n^2)$. In addition, if the classes are known in advance (i.e. the step for computing them can be avoided) then the overall time bound is simply 0(n).

REFERENCES

[1] J.Blazewicz and K.Ecker, A Linear time algorithm for restricted bin packing and scheduling problems. Oper. Res. Letters 2 (1983) 80-83;

[2] S.S.Cosmadakis and C.H.Papadimitriou, The traveling salesman problem with many visits to few cities. SIAM J. Comp. 13 (1984) 99-108;

[3] G.Dantzig, D.R.Fulkerson and S.Johnson, Solution of a large scale traveling salesman problem. Oper. Res. 2 (1954) 393-410;

[4] H.W.Lenstra, Jr., Integer programming with a fixed number of variables. Maths. for Oper. Res. 8 (1983) 538-548;

[5] H.N.Psaraftis, A Dynamic programming approach for sequencing groups of identical jobs. J. Oper. Res. Soc. Amer. 28 (1980) 1347-1359.