Journal of Industrial Engineering 3(2009)49-58

New Heuristic Algorithms for Solving Single-Vehicle and Multi-Vehicle Generalized Traveling Salesman Problems (GTSP)

Ellips Masehian*

Industrial Engineering Department, Tarbiat Modares University, Tehran, 14155-4838, Iran.

Received 3 jun., 2009; Revised 2 Jul., 2009; Accepted 10 Jul., 2009

Abstract

Among numerous NP-hard problems, the Traveling Salesman Problem (TSP) has been one of the most explored, yet unknown one. Even a minor modification changes the problem's status, calling for a different solution. The Generalized Traveling Salesman Problem (GTSP) expands the TSP to a much more complicated form, replacing single nodes with a group or cluster of nodes, where the objective is to find a minimum-length tour containing exactly one node from each cluster. In this paper, a new heuristic method is presented for solving single-vehicle single-depot GTSP with the ability of controlling the search strategy from conservative to greedy and vice versa. A variant algorithm is then developed to accommodate the multi-vehicle single-depot condition, which is modified afterwards to accommodate the multi-vehicle multi-depot GTSP.

Keywords: Traveling Salesman Problem, Single-Vehicle and Multi-Vehicle Generalized Traveling Salesman Problem.

1. Introduction

Through distribution activities, a commonly encountered situation happens when the distributing agent has to visit a number of customers located in different places, pick-up or deliver entities, and return back to its origin only after paying service to all customers. This problem, widely known as the *Traveling Salesman Problem* (*TSP*), applies to a variety of problems like distributing food or goods among shops or houses by a single vehicle starting and ending its trip from a single depot, collecting students of an area by a school bus, delivery of products to various warehouses from a factory, and many other applications [3].

The TSP can be mathematically described as follows: Let a network $G = [N, A, C]$ be defined with N the set of nodes, A the set of branches, and $C = [c_{ij}]$ the matrix of costs. That is, c_{ij} is the cost of moving from node i to node j. Of course, other metrics such as time and distance can also be considered. The TSP requires finding a *Hamiltonian cycle* in G with minimal total cost (a Hamiltonian cycle is a cycle passing through each node $i \in N$ exactly once); that is, it requires the determination of a minimal cost cycle that passes through each node in the relevant graph exactly once.

*Corresponding author, Email: masehian@modares.ac.ir

If costs are symmetric, i.e. the cost of traveling between two locations does not depend on the direction of travel, the TSP is called *symmetric* or *undirected*; otherwise, *asymmetric* or *directed*. A feasible solution to a symmetric TSP has two arcs incident to each node, whereas for an asymmetric TSP there is one arc into and one arc out of every node.

It is shown that the traveling salesman problem belongs to the NP-hard class of problems [11]. This remains true even when additional assumptions such as the triangle inequality or Euclidean distances are invoked [6, 16]. These results imply that a polynomially-bounded exact algorithm for the TSP is unlikely to exist.

As a result, due to the difficulty of the TSP, many heuristic procedures have been developed as a number of early works [4, 9, 13], though most of them encounter problems with storage and running time for cases with more than about 100 nodes.

One of the simplest heuristics for the TSP is the so-called *Nearest Neighbor Heuristic*, which attempts to construct Hamiltonian cycles based on connections to near neighbors, as explained in Fig. 1.

The nearest neighbor procedure for a standard TSP runs in time $O(n^2)$. No constant worst-case performance guarantee can be given. In fact, it can be shown that for arbitrarily large n there exists TSP instances on n nodes

such that the nearest neighbor solution is $\Theta(\log n)$ times as long as an optimal Hamiltonian cycle. This also holds if the triangle inequality is satisfied [19].

If one displays nearest neighbor solutions, the reason for this poor performance can be realized. The procedure proceeds very well and produces connections with short edges in the beginning. Observing the graphical display of a typical solution, we will see that several nodes are 'forgotten' during the algorithm's execution and have to be inserted at high cost at the end.

Procedure Nearest Neighbor Heuristic

10 Select an arbitrary node j
20 Set $l = j$ and $W = \{1, 2, ..., n\} \setminus \{j\}$
30 While $W \neq \varnothing$ do
40 Let $j \in W$ such that $c_{lj} = \min\{c_{li} \mid j \in W\}$
50 Connect l to j and set $W = W \setminus \{j\}$ and $l = j$.
60 Connect l to the node selected in 10 to form a Hamiltonian cycle.

Fig. 1. Pseudo code for the Nearest Neighbor Heuristic.

1.1. Generalizing the Traveling Salesman Problem

The original TSP can be modified in very different ways, which may result in more specific or more general problems [2], as follows:

1) Modification of the size of available fleet to:
   - one vehicle
   - Multiple vehicles (MTSP).
2) Modification of the type of available fleet to:
   - homogenous (only one vehicle type)
   - heterogeneous (multiple vehicle types)
   - Special vehicle types (compartmentalized, etc.).
3) Modification of the housing of vehicles to
   - single depot (domicile)
   - Multiple depots.
4) Modification of the nature of demands to:
   - deterministic (known) demands
   - stochastic demand requirements
   - Partial satisfaction of demand allowed.
5) Modification of the location of demands to:
   - at nodes (not necessarily all)
   - on arcs (not necessarily all)
   - Mixed.
6) Modification of the underlying network to:
   - undirected (symmetric)
   - directed (asymmetric)
   - mixed
   - Euclidean.
7) Modification of the vehicle capacity restrictions to:
   - imposed (all the same)
   - imposed (different vehicle capacities)
   - Not imposed (unlimited capacity).
8) Modification of the maximum route times to:
   - imposed (same for all routes)
   - imposed (different for different routes)
   - not imposed.
9) Modification of the operations to:
   - pick-ups only
   - deliveries only
   - mixed
   - Split deliveries (allowed or disallowed).
10) Modification of the costs to:
   - variable or routing costs
   - fixed operating or vehicle acquisition costs
   - Common carrier costs (for un-serviced demands).
11) Modification of the objectives to:
   - minimize total routing costs
   - minimize sum of fixed and variable costs
   - minimize number of vehicles required
   - maximize utility function based on service or convenience
   - Maximize utility function based on customer priorities.

In addition to these numerous generalizing methods, another type of generalization is still possible:

Suppose that the set of all nodes, N, is decomposed into m independent subsets or clusters. Each subset S_i, contains n_i nodes, such that for $\forall\, i, j \in \{1, \ldots, m\}$, $i \neq j$:

$$S_i \cap S_j = \varnothing, \qquad \left| \bigcup_{i=1}^{m} S_i \right| = \sum_{i=1}^{m} n_i = N$$

The nodes are connected by a network of edges in a way that

$$S_k \neq S_l; \qquad \forall\, i \in S_k,\ \forall\, j \in S_l,\ (i, j) \in E,\ \ i \neq j,$$

Where E is the set of network edges.

Assuming that $C = [c_{ij}]$ is the cost matrix of traversing an edge, the goal is to find a minimal cost cycle (tour) which passes through all clusters and selects only one node to visit within each cluster. This is a *Generalized Traveling Salesman Problem* (GTSP), and can be converted to TSP by setting $m = N$.

Let us provide another definition for the GTSP: Given a graph $G = (V, E)$, of $N \geq 3$ nodes together with distances (weights) for the elements of E (the nodes and the edges of G are the ground set of elements). The set of N nodes (the set V) is partitioned into $m \leq N$ nonempty subsets and the 'traveling salesman' has to visit each subset exactly once. That is, he visits exactly one node is each subset.

Thus, for the GTSP, the optimal solution forms a distance minimizing circuit (if one exists) of exactly k nodes with one node from each subset. Similar to the TSP, it is proved that the GTSP is also an NP-hard problem, although in [5] an attempt was made to respond intuitively to a hypothetical question whether the GTSP is harder than the TSP.

1.2. Related Work

The Generalized Traveling Salesman Problem has found wide applications both in traditional and modern fields of science and technology. some well-known applications include: vehicle routing, warehouse order picking, manufacturing, computer operations, examination timetabling, cytological testing, integrated circuit testing, and computer program restructuring [12].

The first work on the GTSP dates back to 1969 and is due to Henry-Labordere [10]. Since then, many researchers have tried to explore the properties of the GTSP and propose efficient solutions for it. A number of works have tried to reformulate the GTSP into more concise forms, such as [17], which provides six compact Integer Programming formulations for the GTSP by using two approaches: the first by using auxiliary flow variables beyond the natural binary edge and node variables, and the second by distinguishing between global and local variables. In [14], a lower bound on the total cost of an optimum solution is computed by employing a Lagrangian relaxation, and an upper bound is heuristically set. These bounds are used to determine and remove those arcs and nodes that are guaranteed not to be in the optimal solution. The problem is then solved by a branch-and-bound procedure.

Another group of existing works have adopted the approach of approximating or simplifying the exact GTSP formulation. For instance, [7] provided an approximation algorithm in which by some problem-specific IP formulation, LP reduction and the max-flow min-cut theorem, a representative set is constructed. In [15], the GTSP is transformed into a standard asymmetric TSP over the same number of nodes. This transformation allows certain routing problems to be modeled using the TSP framework.

The work [8] presents a reduction algorithm that deletes redundant vertices and edges, preserving the optimal solution. The algorithm's running time is $O(n^3)$ in the worst case, but it is significantly faster in practice. The algorithm has reduced the size of GTSP by 15 to 20% on average, resulting in decreased solution times by 10 to 60%. In [1] the possibility of transforming asymmetric GTSP into symmetric TSP is studied and some experimental results are reported.

A variation of the GTSP, called *Probabilistic GTSP* (PGTSP) is addressed in [21], in which each customer belongs to a cluster that consists of a set of customers. Whether or not any given customer will be present during actual operations is known a priori only probabilistically. The work proposes an exact solution algorithm based on the integer L-shaped method and three tour construction-based heuristics for quickly solving the PGSTP.

Among the heuristic approaches for solving the GTSP is [20], which combines a Genetic Algorithm with a local tour improvement heuristic, resulting in optimal and near-optimal solutions. In [18], a composite heuristic for solving the GTSP is proposed which is comprised of three phases: the construction of an initial partial solution, the insertion of a node from each non-visited node-subset, and a solution improvement phase.

In this paper new heuristic algorithms called *Moment-based* algorithms are developed for solving some variations of the Generalized Traveling Salesman Problem: Section 2 deals with the single-vehicle single-depot GTSP, where a detailed solution is provided for a sample problem; Section 3 addresses the multi-vehicle single-depot GTSP; and Section 4 tackles the multi-vehicle multi-depot GTSP. The final Section provides discussion and conclusion.

2. Algorithm for Single-Vehicle Single-Depot GTSP

Although usually rather bad, nearest neighbor solutions have the advantage that they only contain a few severe mistakes. Therefore, they can serve as good starting solutions for subsequently performed improvement methods, and it is reasonable to put some effort in designing heuristics that are based on the nearest neighbor principle.

The *Moment-based* algorithms developed in this paper take advantage of the nearest neighbor heuristic's speed and the rather low cost of connections at the outset of execution. On the other hand, they try to overcome the poor performance of the nearest neighbor method by avoiding the selection of a next visiting point merely based on the closest neighbor. Instead, they make a judgment between *two* closest neighbors, while regarding the consecutive neighbors of those two, and select a neighbor with a minimal total expected cost. An extension to k closest neighbors is discussed in the last Section.

However, in order to apply this concept to a GTSP, where there exist clusters instead of points, we decompose the problem into two repeating phases:

1) Identifying the next cluster to visit,

2) Specifying the node within that cluster to be visited.

Prior to dealing with the algorithm for solving the Single-Vehicle Single-Depot GTSP, some preprocessing steps should be taken, as described below.

2.1. Preprocessing Steps

a) Set $W = \{S_1, S_2, \ldots, S_m\}$, as the set of all clusters.

b) Obtain the Cost Matrix C:

$$C = \{(c_{ij}) \mid \forall\, i, j \in N;\ i, j \notin S_k\}$$

c) Obtain the Distance Matrix D. Any other metric (e.g. time) can also be used instead of distance.

$$D = \{(d_{ij}) \mid \forall\, i, j \in N;\ i, j \notin S_k\}$$

d) Calculate the average cost between each two clusters by:

$$\bar{c}_{S_i,S_j} = \frac{\sum_{i=1}^{n_i}\sum_{j=1}^{n_j} c_{ij}}{n_i \times n_j}, \qquad \forall\, S_i, S_j \in W, \qquad (1)$$

and form the Average Cost Matrix as:

$$\bar{C} = \left\{ \bar{c}_{S_i,S_j} \,\middle|\, \forall\, S_i, S_j \in W \right\} \qquad (2)$$

e) Using the coordinates of each node in each cluster, compute the gravity center of each cluster (i.e. the mean of *x*- and *y*- coordinates of all nodes in the cluster) by:

$$u_{S_k} = \frac{\sum_{i=1}^{n_k} x_{k_i}}{n_k}, \qquad v_{S_k} = \frac{\sum_{j=1}^{n_k} y_{k_j}}{n_k}, \qquad \forall\, S_k \in W \qquad (3)$$

f) Calculate the average distance between clusters by:

$$\bar{d}_{S_i,S_j} = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2}, \ \forall\, S_i, S_j \in W \qquad (4)$$

and form the Average Distance Matrix as:

$$\bar{D} = \left\{ \bar{d}_{S_i,S_j} \,\middle|\, \forall\, S_i, S_j \in W \right\} \qquad (5)$$

g) Calculate the average 'moment' between clusters by

$$x_{S_i,S_j} = \bar{c}_{S_i,S_j} \times \bar{d}_{S_i,S_j} \qquad (6)$$

and form the Average Moment Matrix as:

$$X_{S_i,S_j} = \left\{ x_{S_i,S_j} \,\middle|\, \forall\, S_i, S_j \in W \right\} \qquad (7)$$

h) For each row of the Average Moment Matrix (7), sort the neighbors of each cluster in ascending order, assigning 1 to the closest neighbor, and continuing to rank $m - 1$. Import these rankings in an $m \times m$ matrix called *Cluster Closeness Ranking* (CCR) Matrix.

The algorithm's name '*Moment-Based*' is derived from the fact that it selects the next visiting cluster and node based on the matrix (7), which is the product of *cost* × *distance*, analogous to the momentum of a force in physics, which is the product of *force* × *distance.*

It should be noted that in the absence of the nodes' location or distance data, the Average Cost Matrix (2) can be used instead of the Average Moment Matrix (7).

i) Determine the starting cluster $S_o \in W$ and the starting node P_o within S_o (serving as depot). If not specified, select them arbitrarily. Next, update $W = W \setminus S_o$.

j) Specify a value or a function for the parameter FNP (discussed below).

2.2. Selection Strategy

In order to control and devise a strategy for the advancement of the algorithm, we introduce a control parameter, named *First Neighbor Preference* (*FNP*) rate.

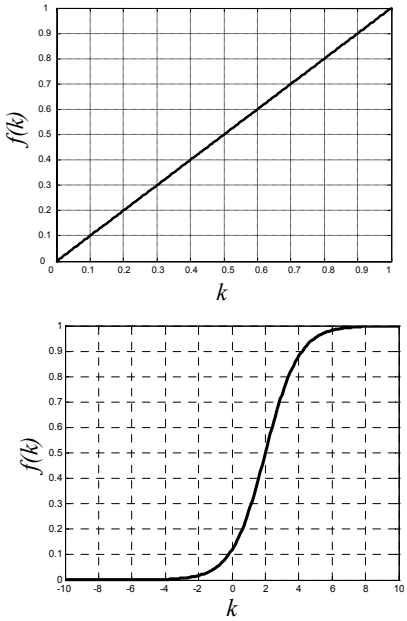
The FNP shows the desire for selecting the nearest cluster as the most appropriate one, thus ignoring the effect of more distant clusters on the current cluster. Higher rates of FNP make the selection greedier and less conservative, i.e. the algorithm adopts a 'short-sighted' attitude towards the selection process. Therefore, adjusting the FNP to a proper value enables the algorithm to work moderately, or even predictive.

We can gain a great flexibility by setting the FNP to a changing variable, rather than a fixed value. For instance, if we let FNP to increase in parallel with iterations progression, the selection trend will be more conservative at the beginning of the process, shifting gradually to a greedy approach, providing some rapid convergence in last iterations.

Moreover, we can control the amount of the increment or decrement of the FNP by assigning a function instead of a constant value, as:

$$FNP_{(k)} = f(k)\,,\ f(k) \in [0, 1] \quad \forall\ k \in [1, N] \qquad (8)$$

where $FNP_{(k)}$ indicates the value of *FNP* at *k*-th iteration, and *N* is the total number of nodes. Different functions may be used to determine the selection strategy dynamically, provided that they keep $FNP_{(k)} \in [0, 1]$, as the linear and sigmoid-type functions displayed in Fig. 3:



$$f(k) = \frac{k}{\|N\|}$$

$$f(k) = \frac{e^{\frac{4k}{\|N\|}-2}}{1+e^{\frac{4k}{\|N\|}-2}}$$

Fig. 3. Some incremental functions for dynamic tuning of the FNP. *N* is the total number of nodes, and *k* is the counter of iterations number.

It is also possible to assign different values or functions to each cluster. For a neither conservative nor greedy approach, set *FNP* = 0.5.

2.3. Main Steps

A) For the starting cluster, S_o, find the first degree neighbor (i.e. the closest cluster) and the second degree neighbor (i.e. the second closest cluster):

$$N_1(S_o) = \left\{ S_j \,\middle|\, x_{S_o,S_j} \le x_{S_o,S_i}, \forall\, S_i \in W \right\}$$

$$N_2(S_o) = \left\{ S_k \,\middle|\, x_{S_o,S_k} \le x_{S_o,S_i}, \forall\, S_i \in W \setminus N_1(S_o) \right\}$$

B) For the clusters $N_1(S_o)$ and $N_2(S_o)$, find each one's closest *unvisited* cluster such that:

$$N_1(N_1(S_o)) = \left\{ S_l \,\middle|\, x_{N_1(S_o),S_l} \le x_{N_1(S_o),S_i}, \forall S_i \in W \setminus N_2(S_o) \right\}$$

$$N_1(N_2(S_o)) = \left\{ S_m \,\middle|\, x_{N_2(S_o),S_m} \le x_{N_2(S_o),S_i}, \forall S_i \in W \setminus N_1(S_o) \right\}$$

C) For determining the next cluster to be visited (S_{new}), check:

$$\begin{aligned} &FNP \cdot x_{S_o,\,N_1(S_o)} + (1-FNP)\cdot x_{N_1(S_o),\,N_1(N_1(S_o))} \le \\ &FNP \cdot x_{S_o,\,N_2(S_o)} + (1-FNP)\cdot x_{N_2(S_o),\,N_1(N_2(S_o))} \end{aligned} \tag{9}$$

If the above condition holds, then $S_{new} = N_1(S_o))$; otherwise, $S_{new} = N_2(S_o))$. This means that S_{new} is a neighboring cluster of the current cluster S_o that has a less moment from S_o. For a better understanding of the above equations refer to Fig. 4.

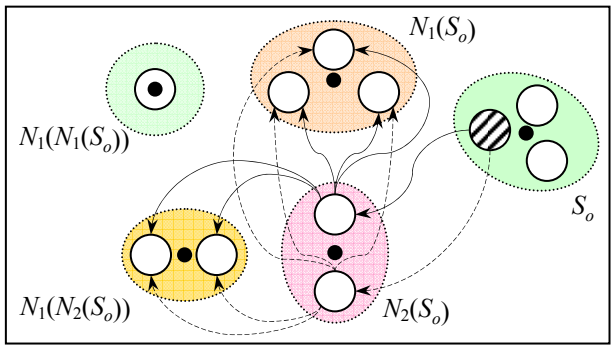


Fig. 4. A representation of relative positions of clusters, with their gravity centers (shown with black solid dots), and containing nodes (white circles). The hatched node signifies the starting node, and the arrows show paths used for determining the next node to visit in Step D.

D) In order to select the next visiting node P_{new} in S_{new}, for each node in S_{new} compute:

$$C_{P_i} = \underset{j}{Max}\left\{ x_{P_o,P_i} + x_{P_i,P_j} \right\}, \tag{10}$$

where $P_o \in S_o, \forall P_i \in S_{new}, \ \forall P_j \in \left\{ N_1(S_{new}), N_2(S_{new}) \right\} \in W$.

For instance in Fig. 4, if the hatched node is $P_o \in S_o$ and $S_{new} = N_2(S_o)$, then the solid and dashed arrows outgoing from each node $P_i \in N_2(S_o)$ are used to compute (10).

The next node to be visited is the one with minimal cost:

$$P_{new} = \text{argmin}\left(C_{P_i} \right), \quad \forall\, P_i \in S_{new}. \tag{11}$$

In other words, P_{new} is a node which its maximum integral moment about the last visited node and either of its two closest clusters is the least among all other nodes in S_{new}.

E) Set $S_o = S_{new}$, $P_o = P_{new}$.
Update $W = W \setminus S_o$.
While $W \neq \varnothing$ go to step **A**, else stop.

Like many other heuristics, the moment-based algorithm is also sensitive to the starting point. It is recommended to start with a cluster which has the minimum total closeness ranking, i.e. its columnar sum in the CCR matrix is the least among all clusters. The maximum iterations required to complete the tour is $m - 1$, where m is the total number of clusters.

In order to illustrate the algorithm's performance, its application to a sample problem is provided bewlow.

2.4. A Numerical Example

A set of 12 nodes is partitioned into 6 clusters, as depicted in Fig. 5. The coordinates of nodes are not available, but the traveling Cost Matrix C is given bellow. Find a minimal cost tour such that all clusters are covered once and only one node within each cluster is visited.

C		S_1	S_2		S_3			S_4		S_5	S_6		
		P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}
S_1	P_1		7	6	11	10	9	6	11	5	14	12	8
S_2	P_2	7			11	15	6	22	12	14	30	25	19
	P_3	6			10	10	12	18	24	31	25	17	26
S_3	P_4	11	11	10				20	17	32	41	29	21
	P_5	10	15	10				10	18	51	26	41	25
	P_6	9	6	12				19	27	54	65	25	23
S_4	P_7	6	22	18	20	10	19			10	26	32	47
	P_8	11	12	24	17	18	27			12	24	58	61
S_5	P_9	5	14	31	32	51	54	10	12		11	32	21
S_6	P_{10}	14	30	25	41	26	65	26	24	11			
	P_{11}	12	25	17	29	41	25	32	58	32			
	P_{12}	8	19	26	21	25	23	47	61	21			

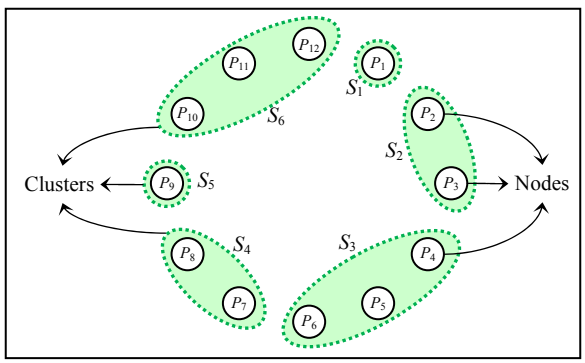


Fig. 5. A schematic of the 12 nodes and 6 clusters in the example. Note that the nodes' locations are not exact.

Preprocessing Steps

a) Set $W = \{S_1, S_2, S_3, S_4, S_5, S_6\}$

b – g) Form the Average Cost Matrix $\overline{C}$. Note that since the data for clusters' distances are unavailable, we can take $\overline{C}$ as the Average Moment Matrix *X*.

$\overline{C}$	S_1	S_2	S_3	S_4	S_5	S_6
S_1		6.5	10	8.5	5	11.33
S_2	6.5		10.66	19	22.5	23.66
S_3	10	10.66		18.5	45.66	32.88
S_4	8.5	19	18.5		11	41.33
S_5	5	22.5	45.66	11		21.33
S_6	11.33	23.66	32.88	41.33	21.33	

h) Form the Cluster Closeness Ranking (*CCR*) matrix:

CCR	S_1	S_2	S_3	S_4	S_5	S_6
S_1		2	3	4	1	5
S_2	1		2	3	4	5
S_3	1	2		3	5	4
S_4	1	4	3		2	5
S_5	1	4	5	2		3
S_6	1	3	4	5	2	

i) Set $S_o = S_1$, $P_o = P_1$, $W = \{S_2, S_3, S_4, S_5, S_6\}$.

j) Set *FNP* = 0.5. Since cluster 1 has the least columnar sum and so is closer to all other clusters, it is appropriately set as the depot (starting point).

Iteration 1

A) $N_1(S_1) = S_5$
$N_2(S_1) = S_2$.
B) $N_1(S_5) = S_4$
$N_1(S_2) = S_3$.
C) $(0.5)5 + (1 - 0.5)11 = 8$
$(0.5)6.5 + (1 - 0.5)10.66 = 8.58$
Next visiting cluster: $S_{new} = S_5$.
D) Node P_9 selected.
E) $S_o = S_5$, $P_o = P_9$, $W = \{S_2, S_3, S_4, S_6\}$.

Iteration 2

A) $N_1(S_5) = S_4$
$N_2(S_5) = S_6$.
B) $N_1(S_4) = S_3$
$N_1(S_6) = S_2$.
C) $(0.5)11 + (1 - 0.5)18.5 = 14.75$
$(0.5)21.33 + (1 - 0.5)23.66 = 22.5$
Next visiting cluster: $S_{new} = S_4$.
D) On node P_7:
Max {(10+22), (10+18), (10+20), (10+10), (10+19)} = 32
On node P_8:
Max {(12+12), (12+24), (12+17), (12+18), (12+27)} = 39
Node P_7 selected.
E) $S_o = S_4$, $P_o = P_7$, $W = \{S_2, S_3, S_6\}$.

Iteration 3

A) $N_1(S_4) = S_3$
$N_2(S_4) = S_2$.
B) $N_1(S_3) = S_2$
$N_1(S_2) = S_3$.
C) $(0.5)18.5 + (1 - 0.5)10.66 = 14.58$
$(0.5)19 + (1 - 0.5)10.66 = 14.83$
Next visiting cluster: $S_{new} = S_3$.
D) On node P_4:
Max{(20+11), (20+10), (20+41), (20+29), (20+21)} = 61
On node P_5:
Max{(10+15), (10+10), (10+26), (10+41), (10+25)} = 51
On node P_6:
Max{(19+6), (19+12), (19+65), (19+25), (19+23)} = 84
Node P_5 selected.
E) $S_o = S_3$, $P_o = P_5$, $W = \{S_2, S_6\}$.

Iteration 4

A) $N_1(S_3) = S_2$
$N_2(S_3) = S_6$.
B) $N_1(S_2) = S_6$
$N_1(S_6) = S_2$.
C) $(0.5)10.66 + (1 - 0.5)23.66 = 17.16$
$(0.5)32.88 + (1 - 0.5)23.66 = 28.27$
Next visiting cluster: $S_{new} = S_2$.
D) On node P_2: *Max* {(15+30), (15+25), (15+19)} = 45
On node P_3: *Max* {(10+25), (10+17), (10+26)} = 36
Node P_3 selected.
E) $S_o = S_2$, $P_o = P_3$, $W = \{S_6\}$.

Iteration 5

A) $N_1(S_2) = S_6$.
Next visiting cluster: $S_{new} = S_6$.
D) On node P_{10}: *Max*{(25+14)} = 39 (distance to node P_1)
On node P_{11}: *Max*{(17+12)} = 29 (distance to node P_1)
On node P_{12}: *Max*{(26+8)} = 34 (distance to node P_1)
Node P_{11} selected.
E) $S_o = S_6$, $P_o = P_{11}$, $W = \varnothing$.

The solution to the problem is therefore the tour:

$$P_1 \rightarrow P_9 \rightarrow P_7 \rightarrow P_5 \rightarrow P_3 \rightarrow P_{11} \rightarrow P_1$$

with a total cost of 64. By choosing other starting nodes or other patterns for the First Neighbor Preference (FNP), alternative solutions may be obtained.

3. Algorithm for Multi-Vehicle Single-Depot GTSP

The Generalized Traveling Salesman Problem discussed so far has only one traveler; that is, the proposed algorithm yields an itinerary for a single vehicle. A more general case is when there are k vehicles ($1 < k <$ number of clusters) available at the depot, each one ready to cover a sub-tour of the whole route, such that all clusters should be visited, again through a single node within. It should be noted that the number of vehicles actually planned could be less that k, in which case some of vehicles may remain unutilized.

The details of an iterative algorithm for solving a multi-vehicle single-depot GTSP are presented below. At each iteration of the algorithm, in addition to determining the next cluster and the node within to be visited, the visiting vehicle is decided as well.

3.1. Preprocessing Steps

a) Calculate the Cost Matrix (C), Distance Matrix (D), Average Cost Matrix ($\bar{C}$), Average Distance Matrix ($\bar{D}$), and Average Moment Matrix (X) as described in the Section 2.1.

b) Determine the starting cluster and the starting node within (serving as depot), S_o and P_o, respectively. If not specified, select them arbitrarily.

c) Taking S_o^k as the last cluster visited by vehicle k, and P_o^k as the last visited node within cluster S_o^k, initially set $S_o^k = S_o$, $\forall\ v_k \in V$, in which V is the set of vehicles.

d) Set $W = \{S_1, S_2, \ldots, S_m\} \setminus S_o$ as the set of clusters available for every vehicle. Note that a cluster can only be visited by a singe vehicle.

3.2. Main Steps

A) Among the unvisited clusters S_j, to determine the next cluster to be visited and the visiting vehicle, for each vehicle v_k calculate a minimal cost for such a trip by:

$$c_j^k = x_{S_o^k,S_j} + x_{S_j,S_o}, \qquad \forall\ S_j \in W \tag{12}$$

$$c_k = \underset{j}{Min}\left\{ c_j^k \right\}, \quad \forall\ j. \tag{13}$$

in which $x_{S_o^k,S_j}$ is the distance between the gravity centers of clusters S_o^k and S_j, obtained from the matrix $\bar{C}$. This means that for every vehicle v_k located on its last visited point, the least cost of moving to a new unvisited point and returning to the depot afterwards is computed. This step guarantees a closed tour for each vehicle.

The next vehicle to move is:

$$v_{new} = \{v_k \mid c_k \le c_i, \forall\ k, i \in V\}, \tag{14}$$

and the next cluster to visit is:

$$S_{new} = \{S_j \mid c_j^{v_{new}} \le c_i^{v_{new}}, \forall\ j, i \in W\}. \tag{15}$$

Update $W = W \setminus S_{new}$.

Note that since we assume the vehicles are identical and are located in the only depot at the beginning, the c_j^k and c_k for all vehicles are equal at the first iteration, and so the first vehicle to move is selected arbitrarily.

B) In order to determine the next node to be visited by the vehicle v_{new}, for each node P_i in S_{new} compute:

$$C_{P_i}^{v_{new}} = \underset{j}{Max}\left\{ x_{P_o^{v_{new}},P_i} + x_{P_i,P_j} \right\} \tag{16}$$

where

$$P_o^{v_{new}} \in S_o^k, \forall P_i \in S_{new}, \forall P_j \in \{N_1(S_{new}), N_2(S_{new})\} \in W.$$

The next node is the one with minimal cost:

$$P_{new} = \operatorname{argmin}\left(C_{P_i}^{v_{new}} \right), \quad \forall\ P_i \in S_{new}. \tag{17}$$

In other words, P_{new} is a node which its maximum integral moment about the last visited node and either of its two closest clusters is the least among all other nodes in S_{new}.

C) Set $P_o^k = P_{new}$
$S_o^k = S_{new}$
while $W \neq \varnothing$ go to step **A**, else stop.

For further clarification of the algorithm's performance, a sample problem is solved below.

3.3. A Numerical Example

Solve a 2-vehicle GTSP for the sample problem introduced in the Section 2.4, assuming that the vehicles start their tours from node P_2 in the cluster S_2 (see Fig. 5).

Preprocessing Steps

a) The matrices $X = \bar{C}$ and CCR are calculated as before.
b) $S_o = S_2,\ P_o = P_2$.
c) $P_o^1 = P_o^2 = P_2, \qquad S_o^1 = S_o^2 = S_2$.
d) $W = \{S_1, S_3, S_4, S_5, S_6\}$

Iteration 1

A) $c_1 = c_2 = Min\{(x_{S_2,S_1}+x_{S_1,S_2}),(x_{S_2,S_3}+x_{S_3,S_2}),(x_{S_2,S_4}+x_{S_4,S_2}),(x_{S_2,S_5}+x_{S_5,S_2}),\ (x_{S_2,S_6}+x_{S_6,S_2})\} = Min\{(\underline{6.5+6.5}),(10.66+10.66),(19+19),(22.5+22.5),(23.66+23.66)\} = 13.$
$v_{new} = v_1$ (arbitrarily), $S_{new} = S_1$, $W = \{S_3, S_4, S_5, S_6\}$.
B) $P_{new} = P_1$ (since it is the only node in S_1).
C) $P_o^1 = P_1,\ P_o^2 = P_2, \quad S_o^1 = S_1,\ S_o^2 = S_2$.

Iteration 2

A) $c_1 = Min\{(x_{S_1,S_3}+x_{S_3,S_2}), (x_{S_1,S_4}+x_{S_4,S_2}), (x_{S_1,S_5}+x_{S_5,S_2}), (x_{S_1,S_6}+x_{S_6,S_2})\} = Min\{(\underline{10+10.66}), (8.5+19), (5+22.5), (11.33+23.66)\} = 20.66$

$c_2 = Min\{(x_{S_2,S_3}+x_{S_3,S_2}), (x_{S_2,S_4}+ x_{S_4,S_2}), (x_{S_2,S_5}+x_{S_5,S_2}), (x_{S_2,S_6}+x_{S_6,S_2})\} = Min\{(\underline{10.66+10.66}), (19+19), (22.5+22.5), (23.66+23.66)\} = 21.33$

$v_{new} = v_1$, $S_{new} = S_3$, $W = \{S_4, S_5, S_6\}$.

B) On node P_4 and for $\forall P_j \in \{S_4, S_6\}$:

$C^1_{P_4} = Max\{(x_{P_1,P_4}+x_{P_4,P_7}), (x_{P_1,P_4}+x_{P_4,P_8}), (x_{P_1,P_4}+x_{P_4,P_{10}}), (x_{P_1,P_4}+x_{P_4,P_{11}}), (x_{P_1,P_4}+x_{P_4,P_{12}}) = Max\{(11+20), (11+17), (\underline{11+41}), (11+29), (11+21)\} = 52$

On node P_5 and for $\forall P_j \in \{S_4, S_6\}$:

$C^1_{P_5} = Max\{(x_{P_1,P_5}+x_{P_5,P_7}), (x_{P_1,P_5}+x_{P_5,P_8}), (x_{P_1,P_5}+x_{P_5,P_{10}}), (x_{P_1,P_5}+x_{P_5,P_{11}}), (x_{P_1,P_5}+x_{P_5,P_{12}}) = Max\{(10+10), (10+18), (10+26), (\underline{10+41}), (10+25)\} = 51$

On node P_6 and for $\forall P_j \in \{S_4, S_6\}$:

$C^1_{P_6} = Max\{(x_{P_1,P_6}+x_{P_6,P_7}), (x_{P_1,P_6}+x_{P_6,P_8}), (x_{P_1,P_6}+x_{P_6,P_{10}}), (x_{P_1,P_6}+x_{P_6,P_{11}}), (x_{P_1,P_6}+x_{P_6,P_{12}}) = Max\{(9+19), (9+27), (\underline{9+65}), (9+25), (9+23)\} = 74$

$P_{new} = P_5$.

C) $P_o^1 = P_5$, $P_o^2 = P_2$, $S_o^1 = S_3$, $S_o^2 = S_2$.

Iteration 3

A) $c_1 = Min\{(x_{S_3,S_4}+x_{S_4,S_2}), (x_{S_3,S_5}+x_{S_5,S_2}), (x_{S_3,S_6}+x_{S_6,S_2})\} = Min\{(\underline{18.5+19}), (45.66+22.5), (32.88+23.66)\} = 37.5$

$c_2 = Min\{(x_{S_2,S_4}+ x_{S_4,S_2}), (x_{S_2,S_5}+x_{S_5,S_2}), (x_{S_2,S_6}+x_{S_6,S_2})\} = Min\{(\underline{19+19}), (22.5+22.5), (23.66+23.66)\} = 38$

$v_{new} = v_1$, $S_{new} = S_4$, $W = \{S_5, S_6\}$.

B) On node P_7 and for $\forall P_j \in \{S_5, S_6\}$:

$C^1_{P_7} = Max\{(x_{P_5,P_7}+x_{P_7,P_9}), (x_{P_5,P_7}+x_{P_7,P_{10}}), (x_{P_5,P_7}+x_{P_7,P_{11}}), (x_{P_5,P_7}+x_{P_7,P_{12}})\} = Max\{(10+10), (10+26), (10+32), (\underline{10+47})\} = 57$

On node P_8 and for $\forall P_j \in \{S_5, S_6\}$:

$C^1_{P_8} = Max\{(x_{P_5,P_8}+x_{P_8,P_9}),(x_{P_5,P_8}+x_{P_8,P_{10}}), (x_{P_5,P_8}+x_{P_8,P_{11}}), (x_{P_5,P_8}+x_{P_8,P_{12}})\} = Max\{(18+12), (18+24), (18+58), (\underline{18+61})\} = 79$

$P_{new} = P_7$.

C) $P_o^1 = P_7$, $P_o^2 = P_2$, $S_o^1 = S_4$, $S_o^2 = S_2$.

Iteration 4

A) $c_1 = Min\{(x_{S_4,S_5}+ x_{S_5,S_2}), (x_{S_4,S_6}+ x_{S_6,S_2})\} = Min\{(\underline{11+22.5}), (41.33+23.66)\} = 33.5$

$c_2 = Min\{(x_{S_2,S_5}+ x_{S_5,S_2}), (x_{S_2,S_6}+ x_{S_6,S_2})\} = Min\{(\underline{22.5+22.5}), (23.66+23.66)\} = 45$

$v_{new} = v_1$, $S_{new} = S_5$, $W = \{S_6\}$.

B) $P_{new} = P_9$ (since it is the only node in S_5).

C) $P_o^1 = P_9$, $P_o^2 = P_2$, $S_o^1 = S_5$, $S_o^2 = S_2$.

Iteration 5

A) $c_1 = Min\{(x_{S_5,S_6}+x_{S_6,S_2})\} = Min\{(21.33+23.66)\} = 45$

$c_2 = Min\{(x_{S_2,S_6}+x_{S_6,S_2})\} = Min\{(23.66+23.66)\} = 47.33$

$v_{new} = v_1$, $S_{new} = S_6$, $W = \varnothing$.

B) On node P_{10} and $P_j = P_2$:

$C^1_{P_{10}} = Max\{(x_{P_9,P_{10}}+x_{P_{10},P_2})\} = Max\{(11+30)\} = 41$

On node P_{11} and $P_j = P_2$:

$C^1_{P_{11}} = Max\{(x_{P_9,P_{11}}+x_{P_{11},P_2})\} = Max\{(32+25)\} = 57$

On node P_{12} and $P_j = P_2$:

$C^1_{P_{12}} = Max\{(x_{P_9,P_{12}}+x_{P_{12},P_2})\} = Max\{(21+19)\} = 40$

$P_{new} = P_{12}$.

C) $P_o^1 = P_{12}$, $P_o^2 = P_2$, $S_o^1 = S_6$, $S_o^2 = S_2$.

The solution to the problem is therefore the tours:

Vehicle 1: $P_2 \rightarrow P_1 \rightarrow P_5 \rightarrow P_7 \rightarrow P_9 \rightarrow P_{12} \rightarrow P_2$
Vehicle 2: Unused
Total cost = 7 + 10 + 10 + 10 + 21 + 19 = 77.

4. Algorithm for Multi-Vehicle Multi-Depot GTSP

Suppose that the initial location of the vehicles is not bound to a specific depot, and there are multiple depots. So the problem is to determine the initial location of each vehicle, such that it starts and ends a trip from a specific point.

Because the Average Moment Matrix (7) will vary for each vehicle k, it is wise to locate each vehicle close to the gravity center of each cluster of nodes, calculated on the basis of the vehicle's own Cost Matrix. Note that never can two vehicles visit a single cluster, and so the available clusters at the beginning are the non-depot clusters.

Assuming that M_k is the set of clusters visited by vehicle k (including the cluster containing its depot), and N_k is the set of visited nodes, the Average Moment of these points would be

$$\overline{X}^{\,k}_{S_k} = \frac{\sum_i \sum_j x^k_{i,j}}{|M_k|}, \quad \forall i, j \in N_k . \tag{18}$$

in which $|M_k|$ is the number of clusters visited by vehicle k.

The Preprocessing and Main steps of the multi-depot variant of the algorithm are the same as described in the Section 3, except for the Step A, which should be replaced with the following:

A) To determine the next cluster to be visited, and the visiting vehicle, calculate:

$$c_j^k = x_{S_o^k, S_j^k} + x_{S_j^k, \bar{X}_{S_k}^k} \quad (19)$$

$$c_k = \underset{j}{Min}\left\{c_j^k\right\}, \quad \forall j \in W. \quad (20)$$

The next vehicle to move is:

$$v_{new} = \{v_k \mid c_k \le c_i, \forall k, i \in V\}, \quad (21)$$

and the next cluster to visit is:

$$S_{new} = \{S_j \mid c_j^{v_{new}} \le c_i^{v_{new}}, \forall j, i \in W\}. \quad (22)$$

Applying the above modification causes a group of clusters close to the initial location of a certain vehicle to be allocated to that vehicle, thus minimizing the total traveling distance of each vehicle.

5. Discussion and Conclusion

The standard traveling salesman problem is the source of many *NP*-hard problems. An interesting extension to this problem is the GTSP, where the single nodes are replaced with a cluster of them. The heuristic algorithms proposed in this paper suggest efficient and fast solutions to single-vehicle and multi-vehicle GTSP problems by completing the route through a number of iterations and selecting the next cluster, and afterwards the appropriate node within, by a predictive method, thus avoiding a shortsighted approach. The multi-vehicle GTSP algorithm tackles both single-depot and multi-depot GTSPs.

An effective parameter for controlling the searching process is the First Neighbor Preference (FNP), which enables the algorithm to fluctuate between conservativeness and greediness, by varying between 0 and 1. An interesting additional control over the searching strategy is gained through the possibility of tuning the FNP dynamically, such that for instance, it can be set to lower values (nearly 0) to have the algorithm act conservatively, and then be gradually increased (toward 1) to expedite and focus the search.

The time complexity of the preprocessing phase is in $O(n^2)$, n being the number of nodes, due to forming and sorting of the Cost, Distance, and Moment $n \times n$ matrices. The main steps of the single-vehicle algorithm take $O(n)$ time since the total number of iterations is $n - 1$, and at each iteration the number of calculations performed to determine the next cluster and node to visit is in linear time. Therefore, the Total time complexity of the single-vehicle problem will be in $O(n^2)$, which is consistent with that of the nearest-neighbor algorithm. The total number of iterations in the multi-vehicle algorithm is $n - 1$, and the main steps take $O(kn)$ time as k tours must be calculated at each iteration, and so the time complexity of the multi-vehicle algorithm will be also $O(n^2)$.

A variation to the algorithm is obtained by taking a representative node as the gravity center of a cluster, thus obviating the need for calculating the clusters' centers, as done in (3).

In order to increase the 'exactness' of the algorithm and slacken its heuristic nature, we can extend and complicate the calculation of the next visiting cluster beyond the two levels implemented in steps A, B and C of the single-vehicle algorithm (Section 2.3), or the step A in the multi-vehicle algorithm (Section 3.2). That is, instead of just determining the nearest 2 clusters of the current position (based on which the next cluster to be visited is calculated in (9)), we can determine the nearest k clusters as follows:

$$S_{new} = \left\{ S_i \,\middle|\, i = \operatorname{argmin}\left(\sum_{j=1}^{k} FNP_j \cdot x_{S_o, N_j(S_o)} \right) \right\}, \quad (23)$$

$$\sum_{j=1}^{k} FNP_j = 1,$$

in which $x_{S_o, N_j(S_o)}$ is the moment (i.e. cost × distance) of the j-th nearest cluster to the current cluster S_o. This means that S_{new}, the next cluster to visit from the current cluster, is the one having the minimal total moment up to k levels from the current cluster.

The implementation of the algorithm is very easy and straightforward, as featured in the solved problems. Experimental results showed fast performance of the algorithm, and optimal or suboptimal solutions to the many solved problems.

References

[1] Ben-Arieh D., Gutin G., Penn M., Yeo A., and Zverovitch A., Transformations of generalized ATSP into ATSP: Experimental and theoretical study, Operations Research Letters, Vol. 31, pp. 357-365, 2003.

[2] Bodin L., Golden B., Assad A. and Ball M., Routing and scheduling of vehicles and crews – the state of the art, Computers & Operations Research, Vol. 10, No. 2, pp. 63-211, 1981.

[3] Christofides N., Uses of a vehicle routing and scheduling system in strategic distribution planning, Scandinavian Journal of Mat. Admin., Vol. 7, No. 2, pp. 39-55, 1981.

[4] Crowder H. and Pedberg M., Solving large-scale symmetric traveling salesman problems to optimality, Management Science, Vol. 26, No. 5, pp. 495-509, 1980.

[5] Dror M. and Haouari M., Generalized Steiner problems and other variants, Journal of Combinatorial Optimization, Vol. 4, pp. 415-436, 2000.

[6] Garey M., Graham R. and Johnson D., Some NP-complete geometric problems, in Proceedings of the 8th SIGACT Symposium on the Theory of computing, pp. 10-22, 1976.

[7] Glicksman H. and Penn M., Approximation algorithms for group prize-collecting and location-routing problems, Discrete Applied Mathematics, Vol. 156, pp. 3238-3247, 2008.

[8] Gutin, G. and Karapetyan D., Generalized traveling salesman problem reduction algorithms, to appear in Algorithmic Operations Research, arXiv: 0804. 0735v2, 2009.

[9] Held M. and Karp R., The traveling salesman problem and minimum spanning trees, Operations Research, Vol. 18, pp. 1138-1162, 1970.

[10] Henry-Labordere A.I., The record balancing problem: A dynamic programming solution of the generalized traveling salesman problem, RIRO B-2, pp. 43-49, 1969.

[11] Karp R., Reducibility among combinatorial problems, in Complexity of Computer Computations, R. Miller and J. Thatcher (Eds.), Plenum Press, New York, pp. 85-104, 1972.

[12] Laporte G. and Palekar U., Some Applications of the Clustered Traveling Salesman Problem, Journal of the Operational Research Society, Vol. 53, No. 9, pp. 972-976, 2002.

[13] Miliotis P., Integer programming approaches to the traveling salesman problem, Mathematical Programming, Vol. 10, pp. 367-378, 1976.

[14] Noon C.E. and Bean J.C., A Lagrangian based approach for the symmetric generalized traveling salesman problem, Operations Research, Vol. 39, pp. 623-632, 1991.

[15] Noon C.E. and Bean J.C., An efficient transformation of the generalized traveling salesman problem, Technical Report 91-26, University of Michigan, October 1991.

[16] Papadimitriou C., The Euclidean traveling salesman problem NP-complete, Theoretical Computer Science, Vol. 4, No. 3, pp. 237-244, 1977.

[17] Pop P.C., New integer programming formulations of the generalized traveling salesman problem, American Journal of Applied Sciences, Vol. 4, No. 11, pp. 932-937, 2007.

[18] Renaud J. and Boctor F.F., An efficient composite heuristic for the Symmetric generalized traveling salesman Problem, European Journal of Operations Research, Vol. 108, No. 3, pp. 571-584, 1998.

[19] Rosenkrantz D., Sterns R. and Lewis P., An analysis of several heuristics for the traveling salesman problem, SIAM Journal of Computation, Vol. 6, pp. 563-581, 1977.

[20] Snyder L.V. and Daskin M.S., A random-key genetic algorithm for the generalized traveling salesman problem, European Journal of Operational Research, Vol. 174, pp. 38-53, 2006.

[21] Tang H. and Miller-Hooks E., Solving a generalized traveling salesperson problem with stochastic customers, Computers & Operations Research, Vol. 34, pp. 1963-1987, 2007.