Journal of the Operations Research
Society of Japan
Vol. 34, No. 3, September 1991

SELECTION OF RELAXATION PROBLEMS FOR A CLASS OF ASYMMETRIC TRAVELING SALESMAN PROBLEM INSTANCES

Seiji Kataoka
National Defense Academy

Susumu Morito
Waseda University

(Received March 2, 1990; Revised March 25, 1991)

Abstract It is commonly believed that the use of the assignment problem works well when one selects a relaxation problem within the framework of a branch and bound algorithm to solve an asymmetric traveling salesman problem (TSP) optimally. In this paper, we present asymmetric TSP instances found in real-life setting, and show that the above common belief is not necessarily appropriate. Based on the real-life example, a family of asymmetric TSP instances called SLOPE is considered, which are generated by *deforming* arc lengths of standard two-dimensional TSPs on a plane in a specific manner.

For this type of instances, we show that the assignment relaxation yields poor performance, and propose a minimum 1-arborescence relaxation similar to the minimum 1-tree relaxation that has been successfully applied to the symmetric TSPs. In order to make the algorithm more efficient, the proper selection of a root node and the determination of Lagrange multipliers to increase the lower bounds are explored.

Computational experiments with instances SLOPE and also with the real-life instance show that the proposed algorithm gives better computational performance than the algorithm with the assignment relaxation.

1 Introduction

Most papers in the field of mathematical programming refer to applications of the models discussed. Presentation and discussion of computational experiments are often regarded as necessary ingredients of research on algorithm development, and in fact many papers give results of computational experiments. It is rather rare, however, to find papers which present computational results based on real-life instances. Most studies somehow "generate" instances artifically to evaluate algorithms.

Nothing is wrong to use such randomly generated instances, and this would be necessary for the systematic evaluation of proposed algorithms. However, algorithm evaluation could sometimes be based on experiments with "improper" instances. It would be important to perform algorithm evaluation which reflects characteristics of instances which arise in real-life situations.

For those who solve problems in real-life using mathematical programming, there is no question that the proper choice of models and algorithms is crucial. There exist some guidelines which should be followed to come up with a proper choice of models and algorithms, but more information would be desired for their proper choice. This paper considers the popular traveling salesman problem (TSP) for which a huge number of algorithms have been proposed and evaluated, and shows that the commonly believed guideline for selecting a relaxation problem within a branch and bound scheme is not necessarily appropriate. Specifically, the standard guideline for the selection of a relaxation problem recommends to use "the assignment problem if the instance is asymmetric, whereas the minimum 1-tree if symmetric". See, e.g., [6],[7],[9],[11,p.370,p.378,p.392],[12],[13].

This paper presents a practical problem scenario which could be modeled as an asymmetric TSP, and shows that the assignment relaxation does not produce good performance for the particular type of instances, presenting a case where a commonly-believed guideline leads us to an erroneous decision on algorithm design.

It is well-known for combinatorial optimization problems that computational time may vary substantially even for the instances of the identical size, and also that one may encounter exceptionally difficult instances by chance. However, if a specific class of instances is found to be difficult for some reasons, then it is desired to find a proper algorithm to deal with the class of instances. It is shown that a particular type of asymmetric TSP instances could better be solved by the minimum 1-arborescence relaxation which is proposed in this paper.

This paper is organized as follows. Section 2 describes the real-life problem scenario which motivated our study, and then defines a general class of asymmetric TSP instances called SLOPE. Computational results are given which show that the assignment relaxation produces poor performance for the real-life instance. Similar results are presented for a subclass of instances within SLOPE. The next section proposes the use of the 1-arborescence relaxation to solve SLOPE instances efficiently. To obtain stronger lower bounds, 1)the choice of a root node for the 1-arborescence relaxation, and 2)the proper determination of Lagrange multipliers to increase the lower bounds, are explored. Section 4 demonstrates computational results which indecate the effectiveness of the proposed algorithm to solve the difficult class of SLOPE instances and the real-life instance.

In Appendix A, the real-life instance from which the proposed family of asymmetric TSPs originate is given. Appendix B shows a different type of "deformed" 2-dimensional TSPs, for which computational results similar to instances SLOPE can be observed. They imply the existence of a rather large class of asymmetric TSPs in the real world which are some "modifications" of the Euclidean TSPs. Finally, Appendix C gives the description of a heuristic algorithm to find an initial tour.

2 Real-Life Problem and Modeling—Instance SLOPE

2.1 Real-Life problem

Assume that at a factory, there exist several distinct products to be produced. Production of these products will be performed on a machine, and there are two parameters, for example, temperature and product size, that basically determine the production process. The temperature and size specifications are assumed to be mutually independent. As for temperature, it is desirable to go from high to low as much as possible. There is cost incurred when temperature is changed in either directions, but the cost is substantially higher when it is raised. So there is a penalty, so to speak, to raise the temperature. Sizes, on the other hand, may go from large to small, or vice versa, yet, frequent size changes are certainly undesirable, say, due to required setup losses. Thus just sorting products in the decreasing order of temperature specifications would not generally give a good production sequence. In essence, we seek a production sequence so that changes in both temperature and sizes are as "smooth" as possible,with a particular attention given to temperature raises.

One way of formulating this problem is to plot products on a two-dimensional plane with axes corresponding to temperature and size, somehow assign "cost" of moving from product i to j, and finally find a path which visits all points with the minimum cost. Then, this problem can be formulated as the Traveling Salesman Problem (TSP).

With regard to the "cost" of moving between two products (points), it would be reasonable to assume that the basis for the cost assignment is the (Euclidean) distance between the points. For temperature, however, extra penalty is accessed when it goes from low to high. Because of this "directionality", the resultant "cost" matrix becomes asymmetric, and thus we have an instance of an asymmetric TSP:

Minimize $\{C(T) \mid T \in \mathcal{T}\}$
Here $\mathcal{T}$ is a set of tours that visit all the points exactly once and $C(T)$ is the sum of arc cost on a tour T .

This problem originates from a rael-life problem which arose at a steel plant. Note that this type of problm is expected to exist widely for many sequencing problems in productions and other settings. Numerical data of one real-life instance with 35 products are shown in Appendix A.

To solve the particular asymmetric instance, we first applied a branch and bound algorithm based on the assignment relaxation, and the results of computational experiment are given in Table 4-2. Fundamentally, our program is based on a FORTRAN translation of a BASIC program given in Kobayashi [8, pp.107-128]. To make it more efficient and to get stronger lower bounds, at each subproblem, subroutine AP given in Carpaneto, Martello and Toth [5] and the Balas and Christofides bounding procedures [1] are coded and incorporated into the algorithm.

Throughout this paper, time shown corresponds to CPU seconds on APOLLO DOMAIN Series 4000. The 35-product instance required 3030831 branches and the 971223 CPU seconds. The poor performance of the algorithm motivated us to perform this study.

2.2 Modeling and instance SLOPE

We now define a family of Asymmetric TSP (ATSP) instances, referred to as instances SLOPE, which take two-dimensional Symmetric TSPs (STSPs) on a plane and *deform* their distances of a particular direction $p(\geq 1)$ times as described below. To get an intuitive feel, think of a situation where one moves around a ski slope, from which the above name originated. (Refer to Figures 2-1,2-2.)

Points are originally given on a two-dimensional plane (x, y). The x and y coordinates

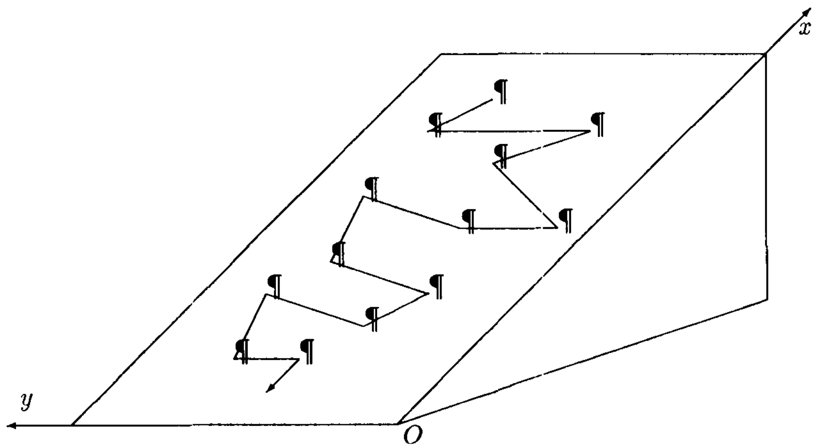


Figure 2-1: A geometric image of instance SLOPE

of point i, (x_i, y_i),$i \in V$, where V is node set, are assumed to be known and are sorted in nondecreasing order of x axe. Assume also that these points are randomly distributed on a square. "Distances" between two points are evaluated as follows:

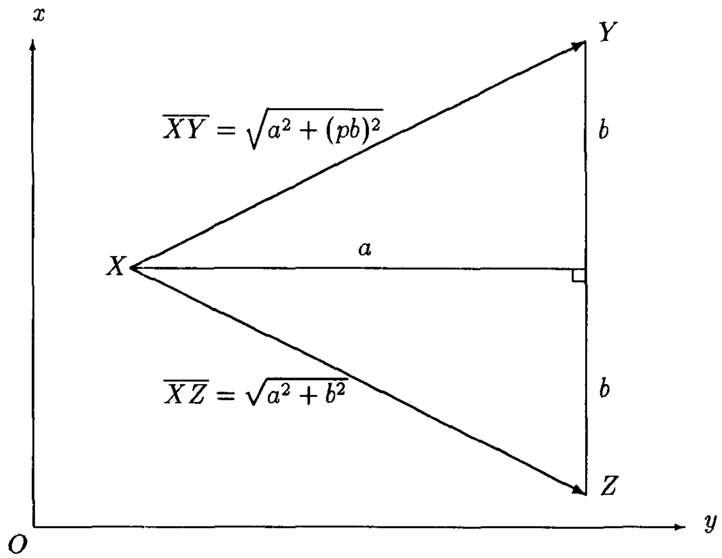


Figure 2-2: Calculation of arc length

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad if \quad x_i \geq x_j \tag{2.1}$$

$$c_{ij} = \sqrt{p^2(x_i - x_j)^2 + (y_i - y_j)^2} \qquad if \quad x_i < x_j \tag{2.2}$$

Note that we "penalize" a move from smaller x_i to larger x_j when $p > 1$. In reality, a starting node was fixed to the node at the top of the slope and it was not required to come back to the starting node. To conform with this practical requirement, we fix the starting node and also add a dummy node D.

Thus, if the original instance consists of $N(N = \mid V \mid)$ products, then the instance to solve becomes an $N + 1$ node problem. For example, the real-life instance with 35 products becomes 36-node instance. Define the lengths of arcs that are incident to D, as in (2.3) and (2.4).

$$c_{Di} = \infty \qquad \text{for all } i \quad \text{except } c_{D1} = 0 \tag{2.3}$$

$$c_{iD} = 0 \qquad \text{for all } i \tag{2.4}$$

Note that node 1 corresponds to the node located at the top of the slope, since nodes are numbered in nondecreasing order of the values of x axe.

2.3 Applications of ATSP algorithms to instance SLOPE

The performance of a standard branch and bound algorithm with the assignment relaxation and with the Balas and Christofides (BC) procedures is examined for instances SLOPE. We parametrically change the number of nodes N (Throughout the paper, N stands for the number of nodes excluding the dummy node.) and slope parameter p, and study their effects on the total number of branches(NB), the ratio of the optimal value of the assignment relaxation of the original problem to its optimal value (AP/OPT), and the ratio of the value of lower bound improved by the BC procedures to the optimal value (BC/OPT), and CPU time. Values of p examined range from 1.0 to 2.0 with an increment of 0.2, together with 3.0, 5.0 and 10.0.

Table 2-1: Experiments for instances SLOPE with the assignment relaxation

N	p	NB	AP/OPT	BC/OPT	CPU
10	1.0	35.85	0.7883	0.9493	0.314
	1.2	34.45	0.7914	0.9357	0.297
	1.4	31.35	0.8002	0.9348	0.275
	1.6	30.80	0.8095	0.9352	0.268
	1.8	28.20	0.8164	0.9321	0.249
	2.0	27.65	0.8210	0.9251	0.248
	3.0	15.45	0.8584	0.9348	0.161
	5.0	5.10	0.9017	0.9545	0.089
	10.0	4.30	0.9237	0.9499	0.080
	RND	7.85	0.9600	0.9711	0.087
12	1.0	97.35	0.8242	0.9543	1.109
	1.2	97.30	0.8314	0.9478	1.097
	1.4	83.85	0.8370	0.9413	0.949
	1.6	69.00	0.8414	0.9380	0.776
	1.8	97.45	0.8468	0.9291	1.054
	2.0	92.00	0.8598	0.9252	0.986
	3.0	37.80	0.8685	0.9174	0.465
	5.0	11.25	0.8988	0.9408	0.194
	10.0	4.30	0.9296	0.9598	0.118
	RND	17.25	0.9471	0.9619	0.215
14	1.0	841.80	0.7987	0.9363	11.808
	1.2	848.45	0.8039	0.9308	12.091
	1.4	1023.95	0.8095	0.9237	14.400
	1.6	890.00	0.8148	0.9203	12.033
	1.8	457.65	0.8170	0.9160	6.447
	2.0	369.95	0.8211	0.9130	5.208
	3.0	134.05	0.8500	0.9185	2.046
	5.0	32.50	0.8784	0.9210	0.593
	10.0	10.30	0.9122	0.9410	0.239
	RND	27.15	0.9487	0.9646	0.427
15	1.0	1063.15	0.8080	0.9333	17.029
	1.2	1161.65	0.8132	0.9252	19.466
	1.4	852.60	0.8192	0.9208	13.731
	1.6	773.05	0.8233	0.9187	12.576
	1.8	757.95	0.8258	0.9127	12.378
	2.0	691.25	0.8297	0.9077	11.280
	3.0	515.65	0.8421	0.8916	8.146
	5.0	99.10	0.8648	0.9073	1.828
	10.0	19.00	0.8921	0.9180	0.445
	RND	61.55	0.9612	0.9702	0.932
16	1.0	4251.75	0.7967	0.9366	80.871
	1.2	4317.10	0.7993	0.9169	79.104
	1.4	30104.75	0.8035	0.9113	520.350
	1.6	35272.50	0.8067	0.9042	607.400
	1.8	25150.55	0.8137	0.8981	444.000
	2.0	18983.65	0.8135	0.8979	333.852
	3.0	723.00	0.8223	0.8950	13.289
	5.0	148.15	0.8460	0.9019	3.074
	10.0	16.80	0.8747	0.9115	0.502
	RND	88.40	0.9696	0.9743	1.478

The preformance of the algorithm on random ATSPs of the corresponding size is also shown for comparisons. Those rows denoted as RND in Table 2-1 give results for $N+1$ node random ATSPs in consideration of the added dummy node for SLOPE instances.

Values of x and y are generated as uniform real random numbers between 1 and 100. Distances between nodes are calculated by (2.1), (2.2) and for the dummy node as defined by (2.3), (2.4) whose fractional parts are truncated to make them integers. With regard to RND, the arc lengths are generated as uniform random numbers between 1 and 100. All figures reflect average results of 20 trials.

Table 2-1 summarizes the results from which the following observations can be made:
1) There is a general tendency that as p increases, NB decreases when p gets greater than a certain value. This is natural because instances become less symmetric as p increases, and because the algorithm tends to work well for totally random and totally asymmetric ATSPs, where each c_{ij} is independent. Note that the maximum NB occurs when p is between 1 and 2. We emphasize the fact that NB is not uniformly decreasing as p increases. This is clearly shown when $N = 16$.
2) The ratio AP/OPT increases almost uniformly as p increases.
3) The ratio BC/OPT deteriorates almost uniformly as p increases for the range of p less than or equal to 2, contributing to the increase of NB as well as CPU.
4) As N exceeds 15, NB increases rapidly.
5) It may be possible to view that instances RND roughly correspond to instances SLOPE with parameter p ranging between 3 and 10, from the standpoint of NB and CPU. Note, however, that with respect to the strength of lower bounds such as AP/OPT and BC/OPT, instances RND always yields stronger (i.e., higher) lower bounds than SLOPE, irrespective of values of p.

The above observations would imply that the guideline which recommends to "use the assignment relaxation for ATSPs" does not necessarily apply to all instances within the wide class of ATSPs. More specifically, one can recognize within a class of SLOPE instances a non-trivial subclass for which the superiority of the assignment relaxation is doubtful.

3 A Proposed Algorithm for Instance SLOPE

3.1 Minimum 1-arborescence relaxation

The computational experience shown in Section 2 indicates that the real-life instance and SLOPE instances presented in this paper have properties similar to those of STSPs. It then is natural to consider the introduction of the minimum 1-arborescence relaxation, i.e., the directed version of the minimum 1-tree relaxation which is known to work well for STSPs [6]. We thus develop in this section a branch and bound algorithm based on the 1-arborescence relaxation for the type of ATSPs which preserve some flavor of symmetry.

There exist few studies on the applications of the 1-arborescence relaxation to solve ATSPs. Smith [12] is the only one known to the authors to study such algorithms. He used random instances for his computational experiments and concluded that the assignment relaxation works better for ATSPs than the 1-arborescence relaxation. It is not clear,however, the assignment relaxation uniformly outperforms the 1-arborescnece relaxation across the wide variety of asymmetric instances.

In Section 3.2, we discuss a method for selecting a root node of the 1-arborescence, and then discuss in Section 3.3 the determination of Lagrange multipliers.

For completeness, we give definitions of some basic terminologies. An arborescence rooted at node q is a spanning subgraph of a directed graph which contains no subtours, exactly one edge directed into every node except q. A value of an arborescence is a sum of its arc lengths, and a numinum arborescence is one whose value is minimum. A minimum

1-arborescence rooted at node q is a minimum arborescence rooted at node q, plus a minimum-length incoming arc to node q.

3.2 Choice of root node for the 1-arborescence relaxation

Generally, any node can be chosen as a root to give a lower bound when the minimum 1-arborescence relaxation is used. It then is natural to select a root node so that the resultant lower bound is higher. We can show that the strongest lower bound can be obtained by selecting node 1, the node at the top of the slope, as the root node.

Theorem 1

The value of a minimum 1-arborescence becomes the largest when node 1 is chosen as the root node under the conditions (2.3) and (2.4).

Proof

First we define the following.

$arb(i)$ = the value of a minimum arborescence rooted at node i

$one(i)$ = the value of a minimum 1-arborescence rooted at node i

Note that the dummy node D is included in each case.

For arbitrary node q $(\neq 1, D)$,$c_{qD} = c_{D1} = 0$ from the definitions of (2.3) and (2.4), and thus

$$one(1) = arb(1) + c_{D1}. \tag{3.1}$$

In $one(1)$, we call the arc which goes into node q as arc a, and define the length of the arc a as $c(a)$. Removing arc a from $one(1)$, produces an arborescence rooted at node q (Figure3-1), which implies

$$arb(q) \leq one(1) - c(a). \tag{3.2}$$

If we call the minimum-length incoming arc to node q as arc b, and define the length of the arc b as $c(b)$, the definition of a minimum 1-arborescence implies

$$one(q) = arb(q) + c(b). \tag{3.3}$$

From (3.2) and (3.3),

$$one(q) \leq one(1) - c(a) + c(b). \tag{3.4}$$

Since $c(b) \leq c(a)$, we obtain

$$one(q) \leq one(1). \tag{3.5}$$

Theorem 1 is valid even when some arcs that are not incident to dummy node D are fixed in a branch and bound procedure, and thus it is best to select node 1 as the root of a 1-arborescence throughout the process of a branch and bound procedure.

3.3 Improvement of lower bounds by the Lagrange relaxation

We now consider, based on Smith [13], how the Lagrange relaxation can be applied to instances SLOPE, where improvement of lower bounds is tried by putting the degree constraints of the 1-arborescence relaxation into the objective function.

To put those relaxed degree constraints into the objective function, we consider an arbitrary multiplier vector $\{\pi_i \mid i \in V\}$, and transform arc lengths c_{ij} into $c'_{ij} = c_{ij} + \pi_i$. The lower bounds L is obtained by subtracting $\sum_{i \in V} \pi_i$ from the value of a 1-arborescence based on

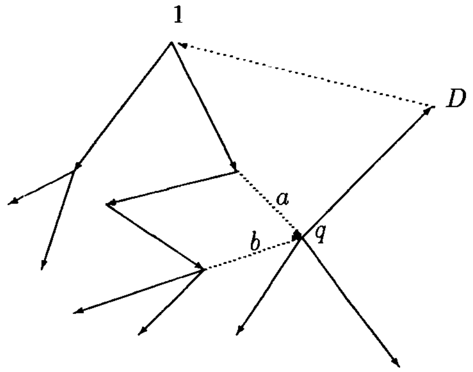


Figure 3-1: Arborescences rooted at node 1 and at node q

the transformed c'_{ij}. This lower bound forms a piecewise linear convex function with respect to$\{\pi_i \mid i \in V\}$, based on which we obtain a new and improved multiplier vector $\{\pi'_i \mid i \in V\}$ by the following formula:

$$\pi'_i = \pi_i + t(d_i - 1) \qquad i \in V. \tag{3.6}$$

Here $d_i(i \in V)$ denotes the out-degree of the 1-arborescence of node i. The step size t can be obtained by:

$$t = \lambda(U - L)/\sum_{i \in V}(d_i - 1)^2 \qquad 0 < \lambda \leq 2, \tag{3.7}$$

where U stands for an upper bound for the value of the optimal tour.

Smith [13] states that the algorithm for the 1-arborescence cannot be used when some arc lengths are negative, and thus forces those Lagrange multipliers which become negative to be 0 as in (3.8):

$$\pi'_i = \max\{0, \pi_i + t(d_i - 1)\} \qquad i \in V. \tag{3.8}$$

Those nodes whose degree is zero should have negative Lagrange multipliers, but because of (3.8) the actual multipliers assigned by Smith's method are zeros. This leads to the solution of the 1-arborescence not increasing the out-degree of those nodes whose associated multiplier should have been negative. The restriction of nonnegativity of arc lengths in the 1-arborescence algorithm is noted also by Suzuki [14]. However, the following simple result holds:

Theorem 2

Given a graph, add a given constant M to all arc lengths. Then a minimum 1-arborescence for the original graph gives the same minimum 1-arborescence for the transformed one, and vice versa.

Proof

Suppose that G is the given graph and G_M a graph where a constant$M$ is added to length of each arc on G. Let T be an arbitrary 1-arborescence, and $z(T)$ the value of T on G. Let T^* be a minimum 1-arborescence. If N denotes the number of nodes in G, any 1-arborescence T consists of N arcs, and thus the value of T^* on graph G_M would be $z(T^*) + MN$. If there exists an optimal minimum 1-arborescence $T^*_M(\neq T^*)$ on G_M, then

(3.9) $$z(T_M^*) + MN \leq z(T^*) + MN,$$

which implies

(3.10) $$z(T_M^*) \leq z(T^*).$$

Unless the inequality in (3.10) holds, however, this contradicts that T^* is a minimum 1-arborescence on G. Therefore, T^* is also the same minimum 1-arborescence on G_M. □

The above theorem shows that even if some arcs have negative lengths, one can add some (possibly big) constant M to all arc lengths to make them nonnegative, and apply a 1-arborescence algorithm. The value of the 1-arborescence is the optimal value obtained for the transformed graph minus MN.

To see the effect of adding constant M instead of forcing negative multipliers to 0 as done by Smith, the following experiments are performed:

1) λ in (3.7) is fixed to 1.25. This is based on the knowledge obtained from the experiments, where various values of λ are tested ranging from 0.25 to 2.0 with a step of 0.25.

2) An initial value of π is set to 0.

3) The termination criteria is as follows:

a. When lower bound exceeds or is equal to an upper bound.

b. Maximum of 100 repetitions if the above condition is not satisfied, where iterations correspond to the number of times π is updated.

With regard to the reason why the number of repetitions is limited to 100, we performed experiments on the real-life instance to see the effect of changing the limit, which showed that the upper limit of 100 repetitions outperforms the other cases with limits ranging from 40 to 160. Refer to computational results given in Table 4-2.

As preliminary computational experiments, lower bounds obtained by the following four cases are compared:

(1) a proposed method based on Theorem 2

(2) the Smith's method

(3) the BC procedures

(4) the assignment relaxation

Evaluations are made based on the ratio between the lower bounds obtained and the optimal value. The instance used is the real-life one in Appendix A. For this instance, the optimal value is known, which is used as an upper bound U in (3.7). The results are shown in Figure 3-2.

The results show that the Smith's method could not improve the lower bounds to the level of the BC lower bound even after 100 iterations, whereas the proposed method dominates the BC lower bound with approximately 20 iterations, and moreover, yields lower bounds which is around 99 % of the optimal value after roughly 80 iterations.

4 Computational Experiments

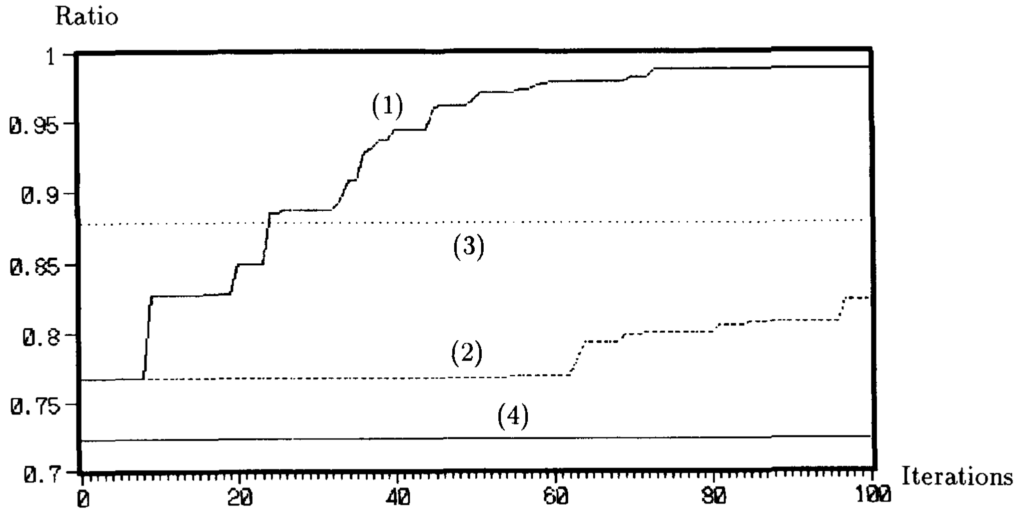
Computational experiments are performed to see the effectiveness of the proposed algorithm. First we describe the results for instances SLOPE, and then for the real-life instance.

4.1 Computational experiments with instances SLOPE

Experiments are performed under the following conditions:

1) A root node is defined to be node 1 as discussed in 3.2.

2) Lower bounds are improved by the Lagrange relaxation as discussed in 3.3. In the algorithm, an initial upper bound is calculated by the heuristic algorithm explained in Appendix C.



(1)proposed method (2)Smith's method (3)BC procedures (4)assignment

Figure 3-2: Increase of lower bounds

3) The branching rule is the same as that of Smith [13].
4) Problem sizes of N =15,20,25 are tested.

Results are shown in Table 4-1. The performance measures shown include the number of branches(NB), the ratio of the value of the minimum 1-arborescence to the optimal value (ARB/OPT), the ratio of the final lower bound after the Lagrange relaxation to the optimal value (LAG/OPT), and CPU time. Moreover, Table 4-1 shows the total number of 1-arborescence problems solved (TN1A) including the number of repetitions needed to increase the lower bounds with the Lagrange relaxation, and the average number of 1-arborescence problems per subproblem (AN1A), where AN1A = TN1A/NB. For comparisons, figures in parentheses provide corresponding results obtained from the algorithm given in Section 2. Dashes (—) indicates the fact that 20 instances could not be solved within the time limit of 3 days.

Following observations are in order from Table 4-1.

1) With respect to NB (i.e., the number of nodes or subproblems in a branch and bound tree), the proposed algorithm uniformly dominates the assignment relaxation with the only exception of $N = 20, p = 10$.

The 1-arborescence relaxation generally yields stronger lower bounds than the assignment relaxation. Specifically, in the range of $p \leq 2$ of instances SLOPE with $N = 15$, the lower bounds generated by the 1-arborescence relaxation (ARB/OPT) are stronger than those by the assignment relaxation (AP/OPT). For N =20 and 25, the similar trend is expected, even though these ratios are unknown because too much computation time was required for these cases. Recall that instances SLOPE with smaller $p(> 1)$ tend to be difficult. More importantly, the final lower bounds as measured by LAG/OPT are uniformly stronger than those generated by the assignment relaxation after the BC procedures. Obviously, this contributes to the superiority of the 1-arborescence relaxation to the assignment relaxation.

On the other hand, one can easily observe that the ratios AP/OPT as well as BC/OPT are very high for instances RND, indicating that instances RND are substantially different from instances SLOPE with even large values of p. This would imply that instances RND fall

Table 4-1: Experiments for instances SLOPE with the 1-arborescence relaxation

N	p	NB		TN1A	AN1A	ARB/OPT	(AP/OPT)	LAG/OPT	(BC/OPT)	CPU	
15	1.0	18.00	(1063.15)	493.90	27.44	0.9023	(0.8080)	0.9797	(0.9333)	18.812	(17.029)
	1.2	25.05	(1161.65)	709.85	28.38	0.8898	(0.8132)	0.9792	(0.9252)	26.202	(19.456)
	1.4	28.15	(852.60)	857.00	30.44	0.8753	(0.8192)	0.9801	(0.9208)	26.891	(13.731)
	1.6	31.40	(773.05)	957.30	30.49	0.8632	(0.8233)	0.9802	(0.9187)	28.313	(12.576)
	1.8	37.70	(757.95)	1193.20	31.65	0.8530	(0.8258)	0.9712	(0.9127)	33.875	(12.378)
	2.0	43.55	(691.25)	1355.60	31.13	0.8427	(0.8297)	0.9685	(0.9077)	36.920	(11.230)
	3.0	34.55	(515.65)	1097.90	31.78	0.8062	(0.8421)	0.9709	(0.8916)	30.186	(8.146)
	5.0	22.00	(99.10)	719.50	32.70	0.7631	(0.8648)	0.9850	(0.9073)	17.709	(1.828)
	10.0	13.80	(19.00)	439.10	31.82	0.7022	(0.8921)	0.9885	(0.9180)	9.908	(0.445)
	RND	50.05	(61.55)	1478.00	29.53	0.7057	(0.9612)	0.8941	(0.9702)	37.270	(0.932)
20	1.0	62.05	(—)	2314.20	37.30	0.9050	(—)	0.9716	(—)	141.293	(—)
	1.2	67.25	(—)	2507.00	37.28	0.8947	(—)	0.9721	(—)	148.087	(—)
	1.4	86.95	(—)	3754.55	43.18	0.8834	(—)	0.9716	(—)	187.806	(—)
	1.6	141.95	(—)	5275.85	37.17	0.8715	(—)	0.9684	(—)	267.317	(—)
	1.8	176.85	(—)	6658.60	37.65	0.8611	(—)	0.9650	(—)	335.660	(—)
	2.0	192.35	(—)	7392.40	38.43	0.8503	(—)	0.9664	(—)	363.356	(—)
	3.0	103.95	(514550.40)	4184.50	40.25	0.8023	(0.8289)	0.9648	(0.8974)	188.456	(15082.950)
	5.0	43.70	(10234.25)	1779.65	40.72	0.7391	(0.8564)	0.9704	(0.9062)	70.740	(328.350)
	10.0	47.95	(24.30)	1998.50	41.68	0.6616	(0.9027)	0.9820	(0.9319)	67.499	(1.181)
	RND	100.50	(141.85)	3166.60	31.51	0.6873	(0.9449)	0.8652	(0.9514)	133.990	(3.080)
25	1.0	224.85	(—)	8826.15	39.25	0.9016	(—)	0.9739	(—)	835.458	(—)
	1.2	316.80	(—)	12289.35	38.79	0.8895	(—)	0.9696	(—)	1140.250	(—)
	1.4	843.65	(—)	32699.35	38.76	0.8759	(—)	0.9636	(—)	2912.100	(—)
	1.6	1031.45	(—)	40903.60	39.66	0.8606	(—)	0.9621	(—)	3599.900	(—)
	1.8	1438.20	(—)	58622.25	40.76	0.8477	(—)	0.9546	(—)	5030.700	(—)
	2.0	1890.30	(—)	78272.80	41.41	0.8328	(—)	0.9486	(—)	6527.300	(—)
	3.0	1002.75	(—)	46167.95	46.04	0.7796	(—)	0.9460	(—)	3268.900	(—)
	5.0	719.85	(—)	35648.80	49.52	0.7087	(—)	0.9542	(—)	2162.000	(—)
	10.0	168.75	(191.55)	7670.75	45.46	0.6263	(0.8883)	0.9782	(0.9130)	342.518	(12.193)
	RND	168.35	(3390.75)	5933.45	35.24	0.7147	(0.9762)	0.8557	(0.9804)	412.359	(11.248)

into a very special, and peculiar, class of ATSPs.

Note also that values of lower bounds improved by the Lagrange relaxation are relatively independent of the value of the slope parameter p, and give good(i.e., large) lower bounds.

2) These results together with Table 2-1 indicate that, irrespective of the algorithm used, difficult SLOPE instances which require much computational time are those whose parameter values are around $p = 2$. It is interesting to note that as p is increased from 1, instances SLOPE first become more difficult to solve, and as p exceeds 2, they become easier. For those difficult SLOPE instances, the increase of computational time for the proposed algorithm based on the 1-arborescence relaxation is not as substantial as that for the assignment counterpart.

3) As will be expected from the computational experiments given in Section 2, the computational time requirement of the assignment relaxation exploded even for $p = 3$ or 5 when $N = 25$. When N is 20, the assignment relaxation took excessive time for $p \leq 3$. As N increases, the assignment relaxation becomes inefficient even for large values of p.

4) When the assignment relaxation is used for instances SLOPE, NB increases rapidly as N exceeds 15 as shown in Table 2-1. The proposed algorithm, on the other hand, indicates much preferable characteristics in terms of the growth of NB as N increases.

Computational time requirement per subproblem (i.e., a node in a branch and bound tree) is not so heavy when the assignment relaxation is used even with the BC procedures. However, because of the weakness of the resultant lower bounds for instances SLOPE, NB tends to grow rapidly, resulting in an excessive overall computational time.

The 1-arborescence relaxation together with the Lagrange relaxation, on the other hand, requires more time per subproblem, because many 1-arborescence problems must be solved repeatedly as the Lagrange multipliers get modified. The subgradient optimization, however, yields comparatively better lower bounds, which helps reduce NB.

4.2 Computational experiments with the real-life instance

Two branch and bound algorithms, one based on the assignment relaxation with the BC procedures, and the other based on the 1-arborescence relaxation discussed in the present paper, are compared, and the results are shown in Table 4-2.

In Table 4-2, RX denotes the type of relaxation problem to be used, and with regard to

Table 4-2: Experiment for the real-life instance

RX	NB	TN1A	RX/OPT	LB/OPT	CPU
assignment	3030831		0.7231	0.8786	971223.000
1-arb(160)	475	28130	0.7675	0.9493	3149.611
1-arb(140)	475	25330	0.7675	0.9493	2845.613
1-arb(120)	475	22530	0.7675	0.9493	2512.327
1-arb(100)	478	19946	0.7675	0.9493	2242.828
1-arb(80)	547	21309	0.7675	0.9493	2653.280
1-arb(70)	654	25287	0.7675	0.9493	3438.836
1-arb(60)	887	34624	0.7675	0.9403	5163.555
1-arb(50)	3798	155181	0.7675	0.9306	25251.000
1-arb(40)	18705	635209	0.7675	0.9174	93882.000

the 1-arborescence relaxation, the value k in "1-arb(k)" is the maximum number of repetitions allowed for a subproblem in the process of the Lagrange relaxation. Two ratios, RX/OPT and LB/OPT are nothing but AP/OPT and BC/OPT, respectively, for the assignment relaxation (see Table 2-1), and ARB/OPT and LAG/OPT,respectively, for the 1-arborescence relaxation (see Table 4-1).

The results suggests the following observations:

1) The proposed algorithm solves the real-life ATSP instance more than 400 times faster than the algorithm that uses the assignment relaxation. Also, the number of branches required for the assignment relaxation algorithm is more than 6000 times of that for the proposed algorithm. Though the 1-arborescence relaxation takes more time for solving a subproblem, the better improvement of lower bounds contributes more to yield an efficient algorithm.

2) Even within the 1-arborescence relaxation, one can observe the dramatic dependence of the strength of lower bounds on the algorithmic efficiency. A small difference of 0.01 in LB/OPT could change CPU time by one digit. This indicates that the proper selection of the iteration limit k is important for the 1-arborescence relaxation. Setting k too small would likely to hurt the computational efficiency, as seen from Table 4-2.

5 Conclusions and Future Researches

This paper is motivated by a real-life instance of the asymmetric TSP, for which an algorithm based on the assignment relaxation fails to provide good performance, despite a widely believed guideline which recommends to "select the assignment relaxation for ATSPs, whereas the 1-tree relaxation for STSPs".

It is a matter of course that solution algorithms should take full advantages of structure and/or characteristics of the problem of interest. This is especially true for computationally tough (say, $\mathcal{NP}$-hard) combinatorial optimization problems such as the TSP studied in this paper.

It is natural that problem instances reflect characteristics of a real problem scenario. Since information concerning a TSP instance is condensed in its distance matrix, characteristics and structure of the problem would be hidden in the distances. As in the case presented in this paper, even rough ideas known about how distances are "defined" would help identify any potential problem structures. Knowledges of problem structures can be, and should be exploited to determine suitable solution strategy such as the selection of relaxation problems used within the branch-and-bound framework.

For TSP instances of a particular application, the distances (c_{ij}) reflect relative "undesirableness" of going from one node to another. Each node normally corresponds to a particular "entity" in a real world and entities have their own "properties". Distances between two nodes are often defined as some functions of their properties, reflecting the underlying problem structure. This implies that so-called asymmetric TSP instances often used to evaluate ATSP algorithms are very peculiar problem instances in the sense that individual entries in the distance matrix are totally independent of each other, which may be difficult, if not impossible, to find in real world.

The above argument leads us to believe, without proof nor definite evidence, that many of real-life instances of TSPs are some deformations of Euclidean TSPs, without regard to whether they are symmetiric or not. In fact, many such problems are expected, as in our example, to retain flavour of Euclidean TSPs. For much wider range of real-life ATSPs than we anticipate, it is estimated that the 1-tree or 1-arborescence relaxation provides much stronger bounds than the assignment relaxation.

Whether a TSP is symmetric or asymmetric is certainly one characterization of the problem. Standard references (say, [11]) suggest to base the selection of relaxation problem on this symmetric/asymmetric dichotomy. This paper warns that this dichotomy should be taken with caution. Moreover, the discussions given here imply that one should pay more attention, in selecting relaxation problems, to whether or not a problem maintains strong flavour of Euclidean TSPs.

We have not yet come up with definite measures obtained from given distance data of an ATSP nor with a formula that tells us which relaxation problem to choose, even if we restrict our attention to the assignment relaxation and the 1-arboresence relaxation. We guess that independence/dependence of c_{ij}'s play a key factor to come up with some measures. More concretely, the independence/dependence of c_{ij}'s can be measured by, say, 1) the ratio of distances c_{ij} and c_{ji} between two nodes, and 2) relative relationship of nodes. The former gives a local (i.e., pairwise) measure of the independence/dependence, whereas the latter a global measure.

With regard to the local measure, when the variability of these ratios c_{ij}/c_{ji} is higher, distances c_{ij} and c_{ji} can be regarded as more independent. Our results show that for ATSPs with high variability of the ratios, the assignment relaxation dominates the 1-arborescence relaxation.

On the other hand, global measures try to extract some "geometric" structure based on the global and relative relationship of nodes. Recognition of certain forms of "directionality" might lead to a good potential measure. One possibility for recognizing a directionality would take an original (complete) graph, and transform it by removing a longer arc (i.e., an arc with $\max\{c_{ij}, c_{ji}\}$) between each pair of nodes. The sum of in-degrees and out-degrees of each node of the resultant graph would be constant N, where N is the number of nodes. If the out-degrees of the transformed graph are distributed uniformly, i.e., $N, N-1, \ldots, 1$, this would imply the existence of a certain directionality, as this is the case for SLOPE. For instances RND, however, the out-degrees of the tarnsformed graph may not be distributed uniformly between 1 and N.

In the mean time, one quick and dirty, yet easily implementable, test before selecting the type of a relaxation problem would be comparisons of the relative strength of lower bounds, without following blindly unproven criteria such as "use the assignment relaxation for ATSPs, whereas the 1-tree relaxation for STSPs".

In summary, the inherent problem structure is a key factor for selecting a proper solution strategy. More research will be needed to associate problem structures with solution strategies, which allows us to design a mechanism for an "automatic" identification of a proper solution strategy.

Acknowledgements

The authors would like to express their appreciations to the referees and the editor for their comments which greatly improved the readability of this paper. A part of the work of the second author was done while he was visiting School of Industrial Engineering, Purdue University. He would like to express his appreciations to Professors Bruce Schmeiser, Ron Rardin, Don Wagner and Ferdinand Leimkuhler.

References

[1] Balas,E. and Christofides,N., "A Restricted Lagrangean Approach to the Traveling Salesman Problem", *Mathematical Programming*, Vol.21(1981), pp.47-69.

[2] Carpaneto,G. and Toth,P., "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem", *Management Science*, Vol.26(1980), pp.736-7435.

[3] Chu,Y. and Lin,T., "On the Shortest Arborescence of a Directed Graph", *Scientia Sinica*, Vol.4(1965), pp.1396-1400.

[4] Fulkerson,D.,R., "Packing Rooted Directed Cuts in a Weighted Directed Graph", *Mathematical Programming*, Vol.6(1974), pp.1-13.

[5] P., L., Hammer, "Algorithms and Codes for The Assignment Problem",*Annals of Operations Research*, Vol.13(1988), pp.193-223.

[6] Held,M. and Karp,R.,M., "The Traveling Salesman Problem and Minimum Spanning Trees : Part 2", *Mathematical Programming*, Vol.1 (1971), pp. 6-25.

[7] Ibaraki,T.,*Combinatorial Optimization* (in Japanese), San-Gyou Tosho (1983).

[8] Kobayashi,T.,*Network Programming Problem* (in Japanese),Baifukan (1981).

[9] Konno,H. and Suzuki,H., *Integer Programming Problem and Combinatorial Optimization* (in Japanese), Nikka-Giren Shuppan (1982).

[10] Kubo,H. and Okino,N., "An Algorithm for Large Scale Traveling Salesman Problem" (in Japanese), *Journal of the Operations Research Society of Japan*, Vol.19 (1976), pp.272-285.

[11] Lawler,E.,L., Lenstra,J.,K., Rinnooy Kan,A., H.,G. and Shmoys,D.,B.,*The Traveling Salesman Problem*, Wiley (1984).

[12] Smith,T.,H.,C., Srinivasan,V., and Thompson,G.,L., "Computational Performance of Three Subtour Elimination Algorithms for Solving Asymmetric Travelling Salesman Problems", *Annals of Discrete Mathematics*, Vol.1(1977), pp.495-506.

[13] Smith,T.,H.,C., "A LIFO Implicit Enumeration Algorithm for the Asymmetric Travelling Salesman Problem Using a One-Arborescence Relaxation", *Mathematical Programming Study*, Vol.12(1980), pp.108-114.

[14] Suzuki,H.,"A Shortest Path Problem with Visiting Order Constraint",*Proceedings of the 7th Mathematical Programming Symposium of Japan*, (1986), pp.127-142.

A A Real-Life ATSP Problem Instance

Figure A-1 shows the numerical data of a real-life instance.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	9999	0	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999
2	1180	9999	18	14	78	252	296	330	269	288	294	380	370	314	382	384	386	346
3	1180	9999	9999	6	24	198	242	276	281	286	306	326	316	326	328	330	332	343
4	1176	9999	10	9999	40	214	258	292	275	280	300	342	332	320	344	346	348	337
5	1176	9999	19	27	9999	174	218	252	318	311	358	297	292	391	304	306	308	362
6	828	9999	367	375	348	9999	44	78	144	137	184	123	118	217	130	132	134	194
7	740	9999	455	463	436	88	9999	34	100	93	140	79	74	173	86	88	90	150
8	672	9999	523	531	504	156	68	9999	66	59	106	45	40	139	52	54	56	116
9	640	9999	557	547	612	264	176	108	9999	18	24	110	100	45	112	114	116	76
10	620	9999	569	559	606	258	170	102	25	9999	20	74	64	40	76	78	80	57
11	606	9999	603	593	679	331	243	175	45	33	9999	126	116	20	128	130	132	70
12	600	9999	604	612	582	234	146	78	100	62	88	9999	10	112	22	24	26	89
13	592	9999	603	611	584	236	148	80	99	61	87	17	9999	99	12	14	16	76
14	578	9999	640	630	736	388	300	232	83	71	37	174	152	9999	143	145	147	65
15	568	9999	627	635	608	260	172	104	123	85	111	41	24	102	9999	2	4	64
16	564	9999	631	639	612	264	176	108	127	89	115	45	28	106	4	9999	2	62
17	560	9999	635	643	616	268	180	112	131	93	119	49	32	110	8	4	9999	60
18	506	9999	683	673	719	371	283	215	140	114	113	157	135	91	111	107	103	9999
19	498	9999	704	694	768	420	332	264	146	135	112	205	184	90	160	156	152	21
20	494	9999	725	715	821	473	385	317	167	155	122	259	237	84	213	209	205	42
21	480	9999	715	723	695	348	260	192	211	173	199	129	112	190	88	84	80	59
22	460	9999	726	716	756	408	320	252	188	162	161	194	172	139	148	144	140	48
23	452	9999	757	747	833	485	397	329	199	187	154	271	249	132	225	221	217	74
24	450	9999	745	753	726	378	290	222	241	203	229	159	142	220	188	114	110	89
25	420	9999	784	792	762	414	326	258	280	242	268	180	178	259	154	150	146	128
26	400	9999	803	811	781	433	345	277	299	262	287	199	197	279	173	169	165	147
27	398	9999	826	816	932	584	496	428	268	256	223	370	348	185	324	320	316	143
28	390	9999	1083	1049	1447	1099	1011	943	453	476	363	916	863	287	839	835	831	364
29	384	9999	1162	1128	1599	1251	1163	1095	532	554	442	1068	1015	365	991	987	983	443
30	378	9999	1240	1206	1749	1401	1313	1245	610	632	520	1218	1165	443	1141	1137	1133	521
31	372	9999	1318	1284	1900	1552	1464	1396	688	711	599	1369	1316	522	1292	1288	1284	599
32	250	9999	936	926	966	618	530	462	398	372	371	404	382	349	358	354	350	258
33	124	9999	1071	1079	1052	704	616	548	567	529	555	485	468	546	444	440	436	415
34	76	9999	1119	1127	1100	752	664	596	615	577	603	533	516	594	492	488	484	463
35	32	9999	1163	1171	1144	796	708	640	659	621	647	577	560	638	536	532	528	507
36	0	9999	1195	1203	1176	828	740	672	691	653	679	609	592	670	568	564	560	539

	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999	9999
2	344	357	426	372	371	441	470	479	408	530	582	633	684	477	604	628	650	666
3	356	369	372	364	383	387	416	425	420	590	642	693	744	469	550	574	596	612
4	350	363	388	358	377	403	432	441	414	568	620	671	722	463	566	590	612	628
5	399	434	348	385	435	363	387	396	492	834	934	1033	1133	490	526	550	572	588
6	225	260	174	211	261	189	213	222	318	660	760	859	959	316	352	376	398	414
7	181	216	130	167	217	145	169	178	274	616	716	815	915	272	308	332	354	370
8	147	182	96	133	183	111	135	144	240	582	682	481	881	238	274	298	320	336
9	74	87	156	102	101	171	200	209	138	260	312	363	414	207	334	358	380	396
10	69	82	120	83	97	135	164	173	134	279	330	381	433	188	298	322	344	360
11	60	62	172	96	77	187	216	226	114	206	258	309	360	201	350	374	396	412
12	120	155	66	106	156	81	90	99	213	576	676	775	875	211	244	268	290	306
13	107	142	56	93	143	71	95	104	200	542	642	741	841	198	234	258	280	296
14	56	42	187	91	72	202	231	241	93	160	211	262	314	196	365	389	411	427
15	95	130	44	81	131	59	83	92	188	530	630	729	829	186	222	246	268	284
16	93	128	42	79	129	57	81	90	186	528	628	727	827	184	220	244	266	282
17	91	126	40	77	127	55	79	88	184	526	626	725	825	182	218	242	264	280
18	13	26	62	26	40	77	106	115	77	223	275	326	377	131	240	264	286	302
19	9999	13	99	35	27	114	142	152	64	174	226	277	328	140	277	301	323	339
20	20	9999	146	50	31	161	190	200	51	117	168	219	271	155	324	348	370	386
21	78	107	9999	37	87	15	39	48	144	486	586	685	785	142	178	202	224	240
22	49	56	60	9999	19	45	74	83	56	210	262	313	364	105	208	232	254	270
23	52	48	137	31	9999	110	139	149	37	129	181	232	283	124	273	297	319	335
24	108	137	30	37	75	9999	24	33	129	471	571	670	770	127	163	187	209	225
25	147	176	66	76	114	36	9999	9	123	486	586	685	785	121	154	178	200	216
26	166	195	85	95	133	55	19	9999	113	476	576	675	775	111	144	168	190	206
27	121	100	236	100	69	206	190	170	9999	52	104	155	206	112	291	315	337	353
28	288	201	751	333	209	721	736	716	80	9999	12	25	38	333	794	818	840	856
29	367	280	903	412	288	873	888	868	159	20	9999	12	25	403	937	961	983	999
30	445	358	1053	490	366	1023	1038	1018	237	40	20	9999	12	472	1078	1102	1124	1140
31	524	437	1204	568	445	1174	1189	1169	316	61	40	20	9999	541	1220	1244	1266	1282
32	259	266	270	210	217	240	224	204	173	315	358	400	442	9999	103	127	149	165
33	434	463	356	363	401	326	305	285	377	707	798	888	979	153	9999	24	46	62
34	482	511	404	411	449	374	353	333	425	755	846	936	1027	201	48	9999	22	38
35	526	555	448	455	493	418	397	377	469	799	890	980	1071	245	92	44	9999	16
36	558	587	480	487	525	450	429	409	501	831	922	1012	1103	277	124	76	32	9999

Figure A-1: Numerical data of a real-life instance

B Different Versions of "Deformed" Euclidean TSP

If we consider road traffic, we often encounter a situation where going into the center of a city takes more time than going out, thus resulting in asymmetric cost matrix. Since vehicles can be assumed to move on a two-dimensional plane, distances would be same and the resultant "cost" tends to preserve some Euclidean properties, or symmetry.

Consider a "center" on a plane. There is an extra penalty associated with approaching the center, whereas regular Euclidean distance is accessed when going away from the center. This, as opposed to instances SLOPE, may be called instances CONE, as it requires extra efforts or costs to go toward the top of a mountain or a cone. Details of the definition of "distance" between two points i and j on a cone will not be explained here.

The above discussion implies the existence of many TSP problems which are not symmetric, yet closer to symmetric TSPs than those whose arc lengths are totally random. Our computational experiments for other types of "deformed" Euclidean TSPs show that our finding here would apply equally well to other such cases.

Table B-1: Computational experiments with instances CONE

		NB		RX/OPT		LB/OPT		CPU	
N	p	assignment	1-arb	assignment	1-arb	assignment	1-arb	assignment	1-arb
15	1.0	1086.05	15.40	0.7823	0.8744	0.9167	0.9988	17.3349	16.3490
	1.2	693.95	21.35	0.7858	0.8759	0.9128	0.9956	11.6626	22.4133
	1.4	497.85	19.55	0.8627	0.8411	0.9132	0.9968	8.7810	18.3576
	1.6	364.10	23.20	0.7887	0.8230	0.9126	0.9924	5.9230	20.1919
	1.8	263.00	29.25	0.7906	0.8057	0.9137	0.9931	4.8492	26.3282
	2.0	232.65	30.90	0.7908	0.7898	0.9104	0.9910	3.5679	26.8591
20	1.0	(—)	56.65	(—)	0.8775	(—)	0.9935	(—)	128.7699
	1.2	(—)	63.40	(—)	0.8584	(—)	0.9919	(—)	139.5196
	1.4	(—)	50.80	(—)	0.8414	(—)	0.9913	(—)	109.2427
	1.6	(—)	79.70	(—)	0.8223	(—)	0.9853	(—)	150.4980
	1.8	(—)	99.65	(—)	0.8048	(—)	0.9829	(—)	189.2531
	2.0	(—)	100.70	(—)	0.7894	(—)	0.9866	(—)	190.8522

C Heuristic Algorithms for Instance SLOPE

Since instances SLOPE are based on the two-dimensional Euclidean TSPs, it appears natural to consider heuristic algorithms originally designed for the Euclidean TSPs. Heuristic algorithms for the Euclidean TSPs can be classified into two categories, namely, 1)tour construction procedures, in which a subtour is gradually expanded to come up with an initial tour, and 2)tour improvement procedures, where a tour is replaced with an another improved tour [11]. In many cases these two procedures are used in combination, as will be done here, too.

Tour construction procedures start with an initial subtour, and consist of two basic steps, a)node selection step which determines a node to be included in the current subtour, and b)node insertion step which decides where in the existing subtour the selected node will be inserted[11]. Repeated applications of a node selection and a node insertion steps would eventually produce a tour.

To come up with a heuristic algorithm for our instances, the following two approaches are considered as candidates for the node selection step:
(1)Farthest Insertion (FI) method, which selects a node farthest away from the current subtour, and
(2)Nearest Insertion (NI) method, which selects a node closest to the current subtour.
FI is known to work well for the Euclidean TSPs. See, e.g., Chapter 7 of [11]. NI, on the other hand, tries to include nodes easier to include in the existing subtour based on the geometric image.(Fig.2-1)

In order to adopt these methods to our asymmetric instances, we introduce the following

concepts of "distance":
(i) Distance between a subtour and a selected node is defined as the minimum of distances between those nodes in the subtour and the selected node.
(ii) Distance d_{ij} between two points i and j which is used in the selection step is calculated by one of the following three methods:

(C.1) $d_{ij} = \min(c_{ij}, c_{ji})$ – choose the shorter distance

(C.2) $d_{ij} = (c_{ij} + c_{ji})/2$ – use the average of the two distances

(C.3) $d_{ij} = \max(c_{ji}, c_{ji})$ – choose the longer distance

Criteria used in the insertion step is to choose the location such that the cost increase is kept minimum.

After an initial tour is obtained based on the framework just outlined, 3-opt will be performed to improve the tour, and the resultant tour will be used as an initial tour.

Table C-1 compares three different methods to come up with "distances" of tours obtained by the heuristic algorithms against the optimal value for the averages of 20 instances with $N = 15$ and $p = 2.0$.

Table C-1 shows that a combination of NI with 3-opt where distances are calculated by (C.2) gives the best heuristic solutions, and thus the initial tours used for our algorithm is generated based on the above combination.

Table C-1: The accuracy of heuristics

d_{ij}	FI	FI+3-opt	NI	NI+3-opt
(C.1)	1.174	1.124	1.079	1.063
(C.2)	1.078	1.057	1.051	1.042
(C.3)	1.073	1.056	1.112	1.061

Seiji KATAOKA : Department of Computer Science, the National Defense Academy, 1-10-20, Hashirimizu, Yokosuka, Kanagawa 239 Japan

Susumu MORITO : Department of Industrial Engineering and Management, Waseda University, 3-4-1, Okubo, Shinjuku, Tokyo 169 Japan