JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 24 No. 1 (2016), pp. 51-66

A Comparison of Ant Colony Optimization and Genetic Algorithm for Solving the Traveling Salesman Problem

Joanna Ochelska-Mierzejewska[1]

[1]*Lodz University of Technology*
Institute of Information Technology
Wolczanska 215, 90-924 Lodz, Poland
joanna.ochelska-mierzejewska@p.lodz.pl

Abstract. *Every company in today's world faces the constant challenge of cost reduction. For distribution and transport service providers, cost cutting appears to be the main operational goal. The successful companies seek to develop a optimal routes for their fleets to minimize the costs and guarantee a timely delivery of the goods.*

With a growing informatization of the industrial world, it is worth considering the use of intelligent systems as a possible way of solving various types of decision problems, which in turn can contribute to the reduction of costs incurred by a company. Such systems enable multidimensional data analysis and to provide information useful in decision making.

The paper investigates the use of the genetic algorithm and the ants colony optimization algorithm as a solution to the travelling salesman problem. It has been shown that both methods provide satisfactory results in solving the problem under examination.

Keywords: *travelling salesmen problem, ant colony optimization, genetic algorithm.*

1. Introduction

One of the key issues associated with logistics services is transport [1]. The integration of transport processes is fundamental to the functioning of the national and international supply chains. Those processes depend largely on the existing infrastructure: rail, road, sea or air [2]. It is the infrastructure that affects the creation of supply chains and determines transport-related costs, which has a direct impact on the quality and costs of logistics services.

The problem of distribution and transport can be classified into the category of information technology logistics. Combinatorial problems which involve the movement and handling of objects, with spatial, logical and time constraints imposed, are difficult to solve [3]. Due to their exponential computational complexity, such problems can be solved with the use of heuristic algorithms.

The basic distribution and transport model is based on the case involving one car and one type of goods [1, 2]. From a logistics perspective, it can be formulated as follows:

The producer is engaged in the production of the same type of goods. The goods are delivered at fixed intervals (for example, every day or once a week) to N customers with a single vehicle (van).

In this model, the delivery times and the distance between the delivery points (customers) are given. Each product is packed in standard containers. The orders placed by particular customers are defined as the number of containers. The vehicle has a capacity defined as the number of containers it can carry.

Given the above information, it is possible to determine:

- the number of tours,
- the goods to be transported in each tour,
- the duration of the tour, item the collection points for each tour,
- the vehicle route.

The above parameters must be set in such a way that the delivery time is as short as possible and the cost of transport as low as possible. Minimizing the time/-cost determines the optimization criterion for a given logistics problem. This is an exemplary representation of the so-called travelling salesman problem.

The paper starts with presenting the general concept of the travelling salesman problem in Chapter 2. Chapters 3 and 4 discuss the concepts of the genetic

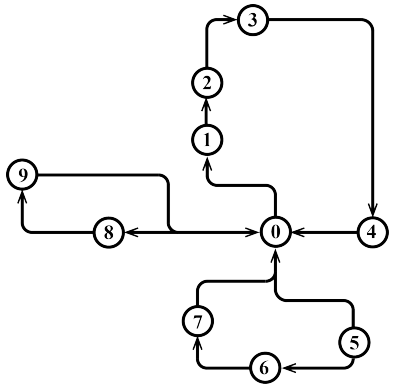

Figure 1: The model of the travelling salesman problem

algorithm and the ant colony optimization algorithm, respectively, which are then applied to the problem under examination. Chapter 5 describes the experiments and discusses the results of a comparison of the two algorithms.

2. The Travelling Salesman Problem

The general concept of the travelling salesman problem (TSP) is as follows: a salesman has to visit n cities (each exactly once) and return to the starting point (the city) [4, 5]. His purpose is to cover a given route at the lowest cost. A sample representation of the problem is shown in Figure 1.

TSP is one of the most popular and most extensively studied topics in computer science and operations research [6]. It is an optimization problem related to the graph theory which consists in finding a Hamiltonian cycle with the lowest weight in the weighted graph G.

This problem can be divided into two basic types: symmetric and asymmetric. In the symmetrical variant, the weighted graph is undirected, which means that the edge between the nodes of the graph in both directions has the same weight. In the asymmetrical variant, the weighted graph is directed — the weight edge for each direction may be different or may not be connected in the second direction.

In the graph theory, the cities in the TPS are the vertices and the roads connecting these cities are edges with specific weights symbolizing, for example, the travel distance, the travel time, and the cost of travel between the cities. Therefore, the task is to determine the Hamilton cycle with the lowest total weight of the

edges contained within the cycle.

A possible solution to this problem consists in searching the whole space of the Hamiltonian cycles in the graph and selecting the one whose sum of weights of all edges is the smallest. The main drawback of this approach is the exponential computational complexity.

The total number of possible Hamiltonian cycles (all routes) for a symmetric problem in the weighted graph is:

$$l = \frac{(n-1)!}{2} \tag{1}$$

where:
l - the total number of Hamiltonian cycles,
n - the number of cities.

The total number of possible Hamiltonian cycles (all routes) for the asymmetric problem in the weighted graph is:

$$l = (n-1)! \tag{2}$$

The travelling salesman problem is NP-hard - there are no known algorithms of a polynomial computational complexity capable of solving it.

In practice, this problem is solved by using approximate algorithms, i.e. those which find the solution approximately equal to the optimum in a short time.

Approximate algorithms are not always able to find the optimal solution. Sometimes, they can return a solution which is much worse than optimal. The application of this type of algorithms is associated with the need to choose between computational speed and solution quality. Usually, it is assumed that the approximate solution is not worse than a certain factor away from the optimal solution / does not differ from the optimal solution more than by a certain value.

3. Ant Colony Optimization Algorithm

The ant colony optimization algorithm is one of the probabilistic optimization methods for solving combinatorial problems [7]. First presented by Marco Dorigo in 1992, it was originally intended to be applied to the problem of finding the optimal path in the graph.

Ant colony optimization falls within the field of intelligent computing. More specifically, it is a class of optimization algorithms based on the principle of swarm

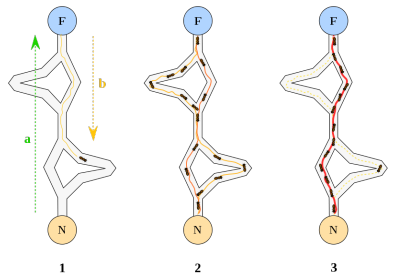

Figure 2: The behaviour of ants foraging for food – pheromone trails
Source: http://chessprogramming.wikispaces.com/Jan+Kozak

intelligence. By definition, an algorithm should always return a correct solution. Since ant colony algorithms are not always capable of finding an optimal solution, they are applied to problems where it is sufficient to find a ”near optimal” solution, which is presented as a set of consecutive steps. Therefore, formally, ant colony optimization algorithms are classified as heuristics.

In the natural environment, ants communicate indirectly by leaving pheromones on the path they follow. An ant foraging for food does not have any knowledge about the surrounding world. It relies on the local perception of its immediate environment. Firstly, the ant moves randomly. If it encounters a food source, it returns to the nest leaving a pheromone trail on the path. Other ants leaving the nest in search for food make decisions about their movement direction on the basis of the pheromone trace left by the previous ants. Of course this choice is conditioned by a certain probability — Figure 2.

The higher the intensity of pheromone, the greater the probability that an ant will select a given path. Each pheromone trail evaporates with time. The strength of the fragrance is affected by two factors:

- the distance from the nest to the food source — on long routes the pheromone fades quickly — the time needed to reach the food source and return to the nest is long. If the distance is small, the pheromone does not have time to evaporate. Thus, the number of ants following that route increases, reinforcing the trail;
- the frequency at which the ants select a specific part of the path — on rarely frequented routes, the pheromone evaporates, whereas on highly frequented routes the trail is reinforced.

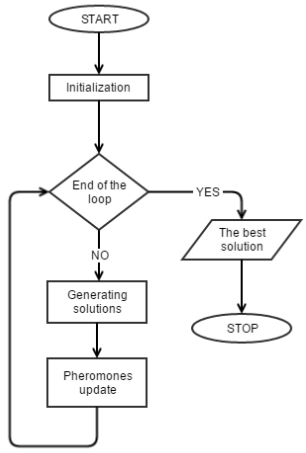

Figure 3: The flowchart illustrating a general scheme of the ant colony optimization algorithm

The discrete search space is the space of permutation of a set of n elements [8]. The general principle of the algorithm is as follows [7, 9]:

1. Initialization:
   - each ant takes a random permutation,
   - the pheromones matrix is filled with the value t_0.
2. Generate solutions [10]:
   - draw the start node,
   - each ant, according to the corresponding probability, selects another node.
3. Compare the solutions obtained and choose the best.
4. Update pheromones according to the best solution.

Figure 3 presents a diagram illustrating the steps of the algorithm.

4. Genetic algorithm

The genetic algorithm is a method for solving optimization problems which mimics the natural evolutionary processes leading to the adaptation of organisms to particular environmental conditions [11, 12].

The task of the genetic algorithm is to search the space of possible solutions via adaptive procedures based on the mechanisms of natural selection and inheritance using the principle of survival of the fittest individuals.

The operation of the classical genetic algorithm is based on duplication of chromosomes and replacement of their subelements [13]. It is composed of the following steps [11, 12, 13, 14, 15, 16]:

1. Initialization — a random selection of a fixed number of chromosomes and the creation of an initial population. Different chromosomes coding methods can be distinguished, e.g. depending on the alignment in the genes or the values stored in genes.

2. Adaptation assessment —- the calculation of an evaluation function for each chromosome.

3. Selection —- the selection of individuals from the current population to the parent population. The individuals characterized by the largest value of fitness function are selected and subjected to genetic operations aimed at creating a new population. There are many methods of selection.

4. Crossover and mutation — the application of genetic operators to a group of chromosomes involving gene chromosomes recombination. The crossover occurs more frequently than a mutation.

5. New population — the chromosomes obtained by crossover and mutation are used to create a new population. The previous population of chromosomes is replaced by a new one, which in the next iteration (generation) becomes the new current population.

6. Derivation of the best individual —- the chromosome with the highest value of the evaluation function is the best solution.

The above steps are presented in Figure 4.

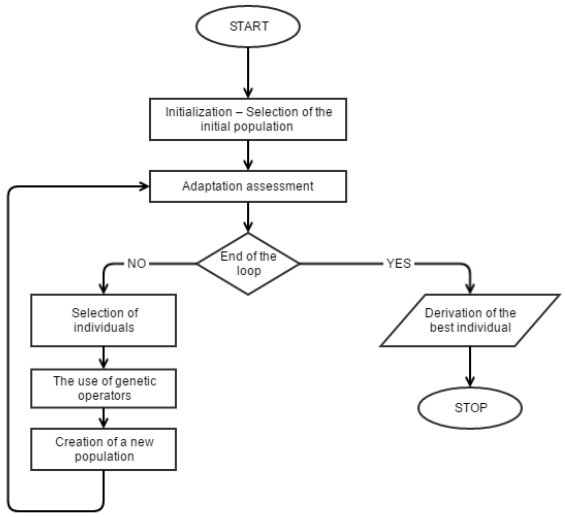

Figure 4: General scheme of the genetic algorithm

5. Experiments

This section presents the results of the experiments and their analysis. More details about the factors for the algorithm parameters applied in this study can be found in [17] for the genetic algorithm and in [18] for the ant colony optimization algorithm. The criteria adopted for both algorithms were as follows:

- input data — the number of random cities equal to 10, 30;
- the parameter values for each algorithm.

The analysis of algorithms includes the number of iterations required, the computation time and the length of the solution (path) obtained.

Each experiment was performed five times. Then, an arithmetic mean value of the number of iterations, the computation time and the path length was calculated. The results are presented in the tables. Cities, as input data, have random coordinates. Therefore, an optimal solution is not known.

The maximum number of 5000 iterations was adopted as the end of the loop. In order to avoid unnecessary iterations and reduce the computation time, the number

of iterations returning the same result were limited to 500.

Moreover, for a more accurate comparison of the results the standard Oliver30 benchmark from TSPLIB was used, for which the optimal solution is 420 [19].

5.1. Experiment 1 for Ant Colony Optimization Algorithm and Random Data

Determination of the appropriate parameters for the ant colony optimization algorithm is not a simple task. It often happens that a given set of parameters is suitable for a certain set of data, whereas for another one the results are far from optimal.

In order to determine the appropriate parameters, a series of tests were performed for the specified data sets. The following experiment was based on the optimized parameters returning acceptable results: an importance pheromone rate $\alpha = 1$, the weighting edge visibility rate $\beta = 1$, the pheromone evaporation coefficient $\rho = 0.01$. There were two changing variables, namely the number of ants, 10 or 30, and the constant pheromone growth factor Q, which was set as 100 or 500.

As demonstrated by the results in Table 1, the best average solution for 10 cities is equal to 700 units. This result was obtained for all possible combinations of parameters in this experiment. In addition, the exact solutions obtained in every single test did not differ from the average. The only apparent difference was the number of iterations required — the higher the number of ants and the greater the value of Q, the lower the number of iterations.

For 30 cities, the smallest average path length found was equal to 1025 units. The other average solutions found differed from the best average result by about 1%. The smallest number of iterations (237 iterations) was obtained for 30 ants and parameter Q equal to 100 and for 10 ants and Q equal to 500.

5.2. Experiment 2 for Ant Colony Optimization Algorithm and Olivier30 Data

For comparison purposes, the ant colony algorithm was tested using Oliver30 benchmark data set, for which the optimal solution is 420 units. The results of this experiment are summarized in Table reftabExperiment2. The best average result, obtained for 30 ants and value of Q equal to 100, with an average of 330 iterations, was equal to 422 units. With those parameter settings, the ant colony optimization algorithm was able to equal the best solution found so far — 420 units.

Table 1: The results of experiment 1

Algorithm parameters					10 cities			30 cities		
Number of ants	α	β	ρ	Q	Number of iterations	Time [ms]	Path length	Number of iterations	Time [ms]	Path length
10	1	1	0.01	100	43	20	700	484	438	1025
30	1	1	0.01	100	17	56	700	237	958	1027
10	1	1	0.01	500	37	19	700	237	319	1035
30	1	1	0.01	500	17	56	700	252	973	1027

Table 2: The results of experiment 2

Algorithm parameters					Oliver30 — TSPLIB		
Number of ants	α	β	ρ	Q	Number of iterations	Time [ms]	Path length
10	1	1	0.01	100	342	358	428
30	1	1	0.01	100	330	1045	422
10	1	1	0.01	500	410	380	427
30	1	1	0.01	500	391	1118	423

For other configurations (10 ants and $Q = 500$) the average solution deviated from the best average by up to 1.4%.

The parameters of algorithm analyzed in this experiments may be accepted as the optimal parameter setting.

5.3. Experiment 3 for the Genetic Algorithm and Random Data

All experiments of the genetic algorithm were performed with the crossover rate equal to 1, the population size of 50, 100 and 200, and with the variable mutation rate equal to 0.015, 0.1, 0.3, and 0.7.

As shown in Table 3, for a population of 50, it is possible to obtain relatively good results with low values of the mutation factor. For 10 cities, the smallest medium path length of 700 units is obtained, with a mutation rate of 0.1 and 0.3. It should be also noted that at a mutation rate equal to 0.015 the result was also good. It is important that the higher the mutation rate, the greater requirement to increase

Table 3: The results of experiment 3

Algorithm parameters		10 cities			30 cities		
Size of the population	Mutation rate	Number of iterations	Time [ms]	Path length	Number of iterations	Time [ms]	Path length
50	0.015	7	19	707	304	74	1167
50	0.1	34	23	700	945	154	1269
50	0.3	112	28	700	192	76	2346
50	0.7	572	53	708	514	125	2467
100	0.015	6	39	702	105	115	1118
100	0.1	8	39	700	1359	387	1319
100	0.3	61	48	700	760	285	2182
100	0.7	425	89	706	804	329	2438
200	0.015	7	77	700	119	238	1103
200	0.1	8	81	700	1450	822	1246
200	0.3	151	114	700	159	300	2233
200	0.7	263	148	700	269	392	2406

the number of iterations. The lowest average number of iterations equal to 7 was obtained at the mutation rate equal to 0.015. For 30 cities, the shortest path, with a length of 1167 units, was achieved at the mutation rate equal to 0.015. The higher the mutation rate, the worse the result. As for the average number of iterations, the results are different than those obtained for the 10 cities. For the shortest path, the algorithm needed 304 iterations; for the path about 100 units longer the algorithm required an average of 945 iterations.

Table 3 shows the beneficial impact of increasing the size of population from 50 to 100 on the average length of the solution, with the mutation rate of 0.015. However, with larger values of the mutation rate, the results deteriorated slightly. The best solutions obtained for 10 and 30 cites, were 700 and 1118 units, respectively. The results were better than those obtained for a population of 50. For 10 cities, fewer iterations were required to achieve the reported results. For 30 cities, the number of iterations decreased only for the mutation rate equal to 0.015. A higher mutation rate increased the computational requirements of the algorithm.

When the population size was increased to 200, further improvements could

be observed in almost all of the specified configuration parameters. The only exception were the results obtained for 30 cities at the mutation rate of 0.3 — for 30 cities the result was worse than that of the experiment performed on a population of 100. The number of iterations needed to obtain the results presented was varied. For small values of the mutation rate (0.015 and 0.1) the number of iterations required was slightly higher. If the mutation rate was equal to 0.3, the number of iterations required was a little higher for 10 cities, whereas for 30 cities this number decreased. For the mutation rate equal to 0.7, the number of iterations decreased in each case. Consecutively, for 10 and 30 cities the best solutions were obtained, namely 700, 1103 and 1559 units, as compared to the results obtained from a population of 50.

5.4. Experiment 4 for the Genetic Algorithm and Olivier30 Data

Experiment 4 was aimed at exploring the performance of the genetic algorithm for a known data set, namely Oliver30 from TSPLIB. The results are summarized in Table 4. The reference point was the optimal solution for Oliver30, which amounts to 420 units.

The best average score was obtained for a population of 200 and the mutation rate equal to 0.015 — 471 units. Note that this result was obtained with the smallest number of iterations. For this population the best exact result was also achieved — 436 units in 41 iterations.

After increasing the mutation rate to 0.1, the average solution improved slightly for small populations and deteriorated slightly for the largest population tested — the best average path length for this mutation rate was 492 units, with a population of 50. Other average results did not differ much — 493 units for a population of 100 and 501 units for a population of 200.

After increasing the mutation rate to 0.3, the average solution deteriorated significantly — the best average path length was 791 units for a population of 50. Although the number of iterations decreased, the results were not satisfactory.

An increase of the mutation rate caused a slight deterioration of results, which was accompanied by a considerable increase in the number of iterations required.

5.5. Comparison of Algorithms

After comparing both algorithms, it was found that the best results were obtained by the ant colony optimization algorithm, which also usually required a

Table 4: The results of experiment 4

Algorithm parameters		Oliver30 — TSPLIB		
Size of the population	Mutation rate	Number of iterations	Time [ms]	Path length
50	0.015	538	103	503
50	0.1	1703	219	492
50	0.3	547	114	791
100	0.015	372	160	495
100	0.1	1668	441	493
100	0.3	452	243	827
200	0.015	218	266	471
200	0.1	1863	976	501
200	0.3	532	462	821

smaller number of iterations. An increase in the size of the population led to obtaining better results by both algorithms.

The best medium-length paths:

- 10 cities:
  - the ant colony optimization algorithm: 700 units,
  - the genetic algorithm: 700 units.
- 30 cities:
  - the ant colony optimization algorithm: 1025 units,
  - the genetic algorithm: 1167 units.

Figures 5 and 6 show the shortest paths in a graph form as designated by the algorithms.

6. Summary

Every company struggling with economic problems tries to reduce the operational costs and, consequently, increase its competitiveness. Multi-dimensional data analysis, forecasting and delivery of critical business information are the key

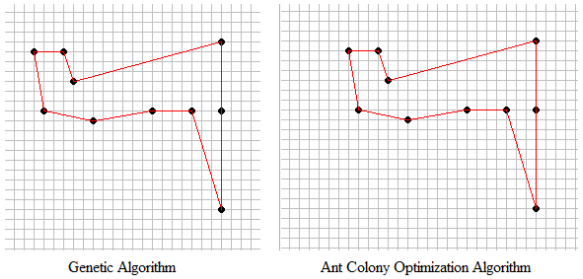

Figure 5: The result of the algorithms in the graph form for 10 random cities

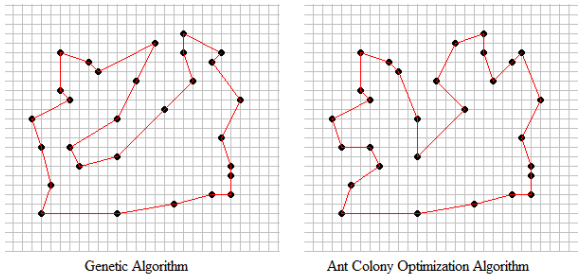

Figure 6: The result of the algorithms in the graph form for 30 random cities

processes that underpin effective decision making and provide insight into certain trends. As a result, a company is able to reduce the costs and become more competitive.

The analysis of a constantly increasing amount of data is a challenge faced by many companies today. It requires the use of information systems which provide the ability to process large volumes of data in a desired manner.

As demonstrated by the results of this study, decision support systems can be very useful in solving the type of transportation and logistics problems under examination. The implementation of the genetic algorithm and the ant colony optimization algorithm has shown that the use of information technology is able to satisfy the needs of a logistics company, which searches for an optimal path in the transport process.

From an information technology perspective, the basic model of the transport problem can be represented as the travelling salesman problem. Both algorithms implemented and tested in this paper have proved to be a great tool for determin-

ing the optimal solution to this problem. However, the ant colony optimization algorithm seems to be more applicable to this task, since it provides slightly better solutions. It also outperforms the genetic algorithm as regards the computation time and the number of iterations needed to find the optimal solution. The greatest difficulty of this method lies in the choice of appropriate parameters, which is not a trivial task, often dependable on the input data.

References

[1] Rydzkowski, W. and Wojewódzka-Król, K., *Transport*, PWN, 1997, in Polish.

[2] Szczepaniak, T. E., *Transport Międzynarodowy*, PWE, 1998, in Polish.

[3] Krasucki, Z. E., *Transport i spedycja w handlu zagranicznym*, Wyd. UG, 1997, in Polish.

[4] Wilson, R., *Wprowadzenie do teorii grafów*, PWN, 1998, in Polish.

[5] *Travelling salesman problem*, [Online] http://fds.oup.com/www.oup.com/pdf/oxed/D2.pdf.

[6] *Travelling salesman problem*, [Online] http://www.travellingsalesmanproblem.com/.

[7] Dorigo, M. and Stützle, T., *Ant Colony Optimization*, MIT Press, 2004.

[8] Hammerl, T., *Ant Colony Optimization for Tree and Hypertree Decompositions*, Betreuer/in (nen): N. Musliu; Institut für Informationssysteme, Arbeitsbereich Datenbanken & Artificial Intelligence, 2009.

[9] Grzymkowski, R., Kaczmarek, K., Kiełtyka, S., and Nowak, I., *Wybrane algorytmy optymalizacji. Algorytmy genetyczne. Algorytmy mrówkowe*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, 2008, in Polish.

[10] Brezina, I. and Èièková, Z., *Solving the Travelling Salesman Problem Using the Ant Colony Optimization*, Management Information Systems, Vol. 6, No. 4, 2011.

[11] Gwiazda, T., *Algorytmy Genetyczne. Wstęp do teorii*, PWN, 1995, in Polish.

[12] Wierzchoń, S., *Sztuczne systemy immunologiczne. Teoria i zastosowania*, Akademicka Oficyna Wydawnicza EXIT, 2001, in Polish.

[13] *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*, PWN, 1999, in Polish.

[14] Goldberg, D., *Algorytmy genetyczne i ich zastosowania*, WNT, 1998, in Polish.

[15] Michalewicz, Z., *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, WNT, 2004, in Polish.

[16] Rutkowski, L., *Metody i techniki sztucznej inteligencji*, PWN, 2012, in Polish.

[17] Ochelska-Mierzejewska, J., *Rozwiązanie problemu komiwojażera przy użyciu algorytmu genetycznego*, Logistyka, Vol. 1, 2016, pp. 350–356, in Polish.

[18] Ochelska-Mierzejewska, J., *Algorytm mrówkowy jako metoda rozwiązania problemu komiwojażera*, TTS Technika Transportu Szynowego, Vol. 12, 2015, pp. 1140–1147, in Polish.

[19] *TSPLIB*, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.