A HYBRID HEURISTIC SOLVING THE TRAVELING SALESMAN PROBLEM

Mircea ANCĂU[1] and Liberato CAMILLERI[2]

ABSTRACT: This paper presents a new hybrid heuristic for solving the Traveling Salesman Problem. The algorithm is designed on the frame of a general optimization procedure which acts upon two steps, iteratively. In the first step of the global search, a feasible tour is constructed based on insertion approach. In the second step the feasible tour found at the first step, is improved by a local search optimization procedure. The second part of the paper presents the performances of the proposed heuristic algorithm, on several test instances. The statistical analysis shows the effectiveness of the local search optimization procedure, in the graphical representation.
KEY WORDS: heuristic, symmetric TSP, optimization.

1 INTRODUCTION

The Traveling Salesman Problem (TSP) plays an important role in the field of combinatorial optimization NP-hard problems. As a discrete optimization problem, the TSP is very difficult to solve optimally in polynomial time, despite its simplicity. It can be stated as follows: given n cities and an $(n \times n)$ matrix d_{ij} whose elements denote the distance between each pair i and j of the n cities, a tour is defined as a closed path which traverses each city exactly once. The problem is to find the path of minimum length i.e. the hamiltonian path. However solving the TSP as an NP - hard problem is not impossible, even if such an algorithm that may solve any of these problems to optimal does not exist (Goertzel, 1993). Only an approximate solution can be expected, which is not too far from the true optimum. Many researchers consider that the existence of such algorithm is undecided in the sense of the Gödel theorem, which states that if neither the existence of such an algorithm cannot be demonstrated, nor can its inexistence. A simple approach to solve the TSP is to list all the possible paths between the cities, then compare all the path lengths to find out which one is the shortest. Unfortunately there are too many paths. For n cities $c_i(x_i,y_i)$, (i = 1, 2, .., n), and the distances between cities obeying the conditions $d(c_i,c_j) \neq 0$ for every $i \neq j$, and $d(c_i,c_i) = 0$, (i = 1, 2, ..., n), the problem is called symmetric if $d(c_i,c_j) = d(c_j,c_i)$, (i = 1, 2, ..., n). The possible paths, also called tours, increase if the number of the cities increases. In the case of the symmetric TSP, for n cities, the possible tour numbers $N_t(n) = (n-1)!/2$. So, for 10 cities there are $N_t(10) = 181,440$ different tours, for 15 cities $N_t(15) = 43,589,145,600$ different tours and so on. When the number of the cities increases, it can easily be seen that exhaustive exploration of all possible tours becomes impracticable, despite the performances of actual computers.

Nevertheless, years of researches in the field of optimization techniques as well as the rapid development of the memory amount and the speed of the computers, have lead to astonishing solutions to these problems. So, in the last 25 years, the record for the most complex combinatorial optimization problem, solved to optimum, increased from 318 cities (Crowder and Padberg, 1980) to 2390 cities (Padberg and Rinaldi, 1987) and 7397 cities (Applegate and Cook, 1993). This last result required the equivalent of at about 4 years CPU time on a SPARC station (Johnson and McGeoch, 1997). Today, problems of 100 cities are solved in seconds or minutes, while problems of 1000 cities or more are solved within hours or days (Padberg and Rinaldi, 1991), (Grötschel and Holland, 1991), (Applegate and Cook, 1993). Anyway, the TSP remains a very difficult problem to solve exactly. It is not only difficult to find out the solution, but even to know its exact value, the so called Held-Karp lower bound. For test instances not very large, there is a way to calculate the optimum value exactly, by linear programming techniques. For other test instances this lower bound may be numerically estimated (Johnson and McGeoch, 1997). In any case, what we may hope in most of the situations is

[1]Technical University of Cluj-Napoca, Faculty of Machine Building, Dept. of Manufacturing Engineering B-dul Muncii 103 – 105, 400641 Cluj-Napoca, Romania

[2]University of Malta, Faculty of Science, Department of Statistics and Operations Research, Msida MSD06, Malta

E-mail: mircea.ancau@tcm.utcluj.ro; liberato.camilleri@um.edu.mt

an approximate solution only, called Probably Approximately Correct (PAC) solution.

Most of the researchers state that there are two possible approaches for solving combinatorial optimization problems: either an optimization algorithm in the strict sense of the word, yielding a globally optimal solution, or an approximation algorithm yielding a very good or possibly optimal solution. A good start for solving the TSP might be a tour developing method, usually a greedy one, which produces a valid solution using a growing process. Another way is to begin from a valid tour and, using a technique based on local search, to get a better solution. Such solving methods are called heuristics (the word heuristic is an adjective involving thought and investigation as a method of problem solving -from Greek *heuriskein* meaning *to discover*). Concerning the operational principle, the heuristic methods for solving the TSP may be generally divided in two classes called *tour construction heuristics* and *successive augmentation heuristics*.

Within the *tour construction heuristics*, the best known are the Nearest Neighbour, the Greedy, the Clarke-Wright and the Christofides. The Nearest Neighbour heuristic is based on a simple rule which imposes each time, going next to the nearest unvisited location (Rosenkrantz et al. 1977). The tour traverses the cities in the constructed order, returning in the first city after visiting the last one in the list. In the case of the Greedy heuristic, the set of cities is viewed as a complete graph with cities as vertices and the distances between each pair of cities as graph edges.

The tour is the hamiltonian path in the graph (Bentley, 1992). The Clarke-Wright heuristic (Frieze, 1979) starts with a pseudo-tour in which one city serves as a central node, also called hub, and all the remaining cities are connected with the hub by two edges. This pseudo-tour starts from the central node and, after visiting each city, the salesman returns to the central node. The rule of the tour construction allows to bypass the hub if this change does not increase the tour length and also if one city does not become adjacent to more than two cities.

The algorithm of the Christofides heuristic starts by constructing a minimum spanning tree and computing a minimum-length matching on the vertices of odd degrees in the tree. Combining these two trees what can be obtained is a connected graph in which every vertex has even degree (Lawler, 1985). This graph must contain a cycle that passes through each city exactly once. Experimental results, on tour construction heuristics showed on random euclidian instances an average percent excess over the Held-Karp lower bound within 23 ÷ 25% for the Nearest Neighbour, 14.2 ÷ 19.5% for the Greedy, 9.2 ÷ 12.2 for the Clarke-Wright, 9.5 ÷ 9.9% for the Christofides (Johnson and McGeoch, 1997).

The group of *successive augmentation heuristics* includes 2-Opt, 3-Opt as simple tour modifications, k-Opt as a generalization of the previous, Tabu Search with variants, Simulated Annealing with variants, Genetic Algorithms and Neural Network with variants. The basic approach to these heuristics is the iterative improvement of a set of randomly selected feasible solutions. The main steps are:

a) Generation of a pseudo-random feasible solution S that satisfies the criterion *valid tour*;
b) Attempt to find an improved feasible solution S_n by means of some transformation of S;
c) If an improved solution is found, i.e. Length (S_n) < Length (S), then replace S by S_n, repeat from step (b);

If no improved solution can be found, then S is the locally optimum solution. Repeat from step (b) until computation time runs out or answers are satisfactory.

The group of 2Opt and 3Opt heuristics may be viewed as a neighbourhood search process, where each tour has an associated neighbourhood of adjacent tours. The algorithm removes two or three edges, therefore breaking the tour into two or three subtours and, after that, reconnects the path in another way (Croes, 1958). There is one remark to be made here. If we restrict only to 2Opt moves which remove crossings (i.e. those that delete from the tour two edges that intersect at a common point):

a) such moves cannot cause the increase of the tour length (under the euclidean norm) due to the triangle inequality;
b) such moves may introduce new crossings. Although, at most, n^3 moves are sufficient to remove all crossings, where n is the number of the cities in that instance (Johnson and McGeoch, 1997).

The kOpt moves are a generalization of the 2Opt and 3Opt moves. As 3Opt moves do, the kOpt moves may make the escape from a local optimum possible, but the running time is considerably slower.

Tabu Search algorithms are based on the observation that not all locally optimal solutions need be good solutions. This is why it modifies the local optimization algorithm by means of a technique which enables it to escape from the local optimum and continue to search (Glover, 1986), (Knox, 1994). These algorithms are based mainly on 2Opt, but differ from to the nature of the tabu list and the implementation of aspiration level. The Lin-Kernighan algorithm is a generalization of the 3Opt and has much in common with the Tabu Search (Lin and Kernighan, 1973).

The Simulated Annealing algorithms introduce a control parameter of the optimization process, which is similar to the temperature in the metallurgy processes of the annealing of metals. The temperature parameter is gradually reduced from a higher value to zero. In this way, the step by step optimization process suffers some modifications from the random way, in which every movement is accepted and turns into a greedy way where perturbations are forbidden.

The system is transferred from one configuration of higher energy into another one with lower energy and ordered configuration. In the Simulated Annealing (Kirkpatrick, 1983) every improvement is accepted without reserve, while deteriorations are accepted with a probability (Metropolis et al. 1953).

Some variants of the Simulated Annealing algorithms, such as the Threshold Accepting (Dueck and Scheuer, 1990) use a deterministic update rule by which a perturbation is accepted only if the made deterioration does not exceed a threshold value (Moscato and Fontanari, 1990). Another variant solves the TSP based on different types of physical processes, such as the inverse of the diffusion process (Ugajin, 2002).

The idea of using genetic algorithms as optimization approaches is not quite new. The genetic algorithms are models of machine learning, based on the biological evolution of population of individuals. The searching process begins by the generation of k starting solutions $S = \{S_1, \ldots, S_k\}$ which is applied to a local optimization algorithm.

The local optimum solutions found at this step will replace the previous starting solutions in S. The optimization process uses different subsets from S, of different size and, combining them by randomized crossover operations, creates new solutions that reflects the aspects of the previous best solutions, using a selection strategy (Valenzuela and Jones, 1994).

The neural net approaches may be divided in two major groups. The first group is based on commonly applicable algorithms, in which the neurons are organized according to some formulation of the TSP as an integer program (Peterson and Söderberg, 1989), (Xu and Tsai, 1991). The second group is limited to geometric instances and is based on algorithms in which neurons are at first viewed as vertices of a polygon. This polygon extends and deforms iteratively in such a way that this vertices will at last match the cities. The resulting distorted polygon will look finally as a tour. There are two variants of these methods, the elastic net (Peterson, 1990), (Simmen, 1991) and the self-organizing map (Angéniol et al., 1988), (Kohonen, 1988).

2 THE PRINCIPLE OF THE OPTIMIZATION ALGORITHM

Traveling salesman problem faced us up to a combinatorial optimization problem, which has multiple local optimal points and one or maybe more absolute optimal points. There are also too many possible tours and that puts exhaustive exploration of all tours out of the question. In the handling of multiple local optimal points, it is suitable to apply a global search procedure, such as the Monte Carlo method, followed by a local search process in the neighbourhood of the previously found solution. If you want to find out an optimum, the Monte Carlo method requires the trying of many different points from the solution space at random and seeing which one of these is the best.

If you check out a sufficient amount of different points, the best you will find will be almost certainly a reasonable guess at the best overall. At this moment we must make the remark that by different points we do not suggest different tours at random. The tour construction procedure is generally dependent on the order in which cities are inserted. A tour construction procedure in which the cities are inserted at random and in a different random order at each iteration, will follow the Monte Carlo method as a global search procedure.

The first step in the heuristic algorithm is reading the initial data in the form of a tsp file and then building the distance matrix. The distance matrix is the largest memory consumer, but the consume can be optimized taking into account that for the symmetric TSP, the matrix elements are symmetric to the main matrix diagonal. After the step of the construction of the distance matrix, two procedures of tour construction and local search are iteratively executed, until the stopping condition is

fulfilled. The criterion of the iteration number was used as a stopping condition.

```
Read initial data and build distance matrix;
while ( stopping condition not satisfied)
{
// The tour construction as global search procedure
  Generate random permutation of n cities;
  Setup a partial tour with the first three cities from
    the above permutation;
  Insert cities in the tour, based on the condition of
    minimum tour length increasing;

// Local search procedure
  Improve the tour by shifting 2Opt, 3Opt and 4Opt;
  Check and remove crossing segments;
  Calculate the tour length;
  if (actual tour length < previous minimum
                          tour length)
  Minimum tour length = Actual tour length;
}
```

Figure 1. The main steps performed by the heuristic algorithm

2.1 The tour construction algorithm

The (x,y) coordinates of the n cities are stored in an array of structures, together with an integer variable which takes into account the index of cities from the tsp file. In the same structure of cities there is a variable which can take the value zero or one, depending on the fact that the city was visited or not. Parallel to the array of structures of the cities, there is an array of pointers in which structure elements addresses are stored. To speed up the computation, instead of moving structure elements, only pointers are moved.

The first step in the construction of the tour algorithm a random sequence of integers from 1 to n, is generated as a permutation. In ANSI C the *rand()* function returns a value of type int (a two byte quantity on many machines), which must be at most 32767. This can be a serious impediment especially in the Monte Carlo approach. In generating a random instance, the portable random numbers generator *ran1()* of Park and Miller (Press et al,1997) was used, which has a period bigger than 10^8. Once the permutation is generated, its sequence is assigned to the array of pointers in which the addresses of cities structures elements are stored.

The tour construction procedure starts with a subtour made by the first three cities from the permutation order. The rest of the cities, in the random permutation order, will be inserted one by one, in this subtour, in a position done by the minimum increasing cost criterion. In this way, step by step the subtour extends until the complete tour is formed.

```
// the generation of n random permutations
for i = 0 to n
  p[i] = i;
for i = 0 to n
{
  // call the random number generator ran1():
  rin = random integer number, between i to (n-i);
  exchange p[i] with p[rin];
}
// Assign the order of the pointers according to the
array p[i]
        which stores the random permutation
for i = 0 to (n-1)
   pointer of city [i] = the address of city [p[i]];
pointer of city [n] = the address of city [p[0]];
```

Figure 2. The random permutation algorithm

2.2 The local search algorithm

The tour construction procedure described above depends on the order in which the cities are inserted. If we start again the construction of the tour, using another insertion order of the cities the result will be different.

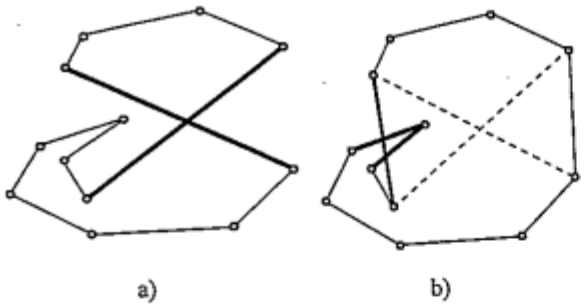
This feature can be exploited as in a Monte Carlo procedure. For a large amount of different cities insertion orders we may retain the best tour variant. But this approach is too slow and the tour constructed at this first step usually has many points of intersection between tour segments. This is not an impediment because we can remove the cross sections and improve the tour. The 2Opt, 3Opt and 4Opt moves were preferred as a local search approaches to the tour.

The 2Opt approach deletes two segments, thus breaking the tour in two paths, and then reconnects those paths in a way that minimizes the tour cost. Any removed crossing leads to a shorter tour, due to the triangle inequality. But it should be remarked that one 2Opt move may introduce new crossings (see Figure 3).

Nevertheless it is known that, at most, n^3 uncrossing 2Opt moves are enough to remove all crossings (Johnson and McGeoch,1997) where n is the number of the cities.

In a similar form, the 3Opt and 4Opt moves delete three or four segments respectively, from the tour, breaking the tour in three or four paths respectively, and then reconnect those paths in the way that also minimizes the tour cost.

There are several ways for breaking the tour and for reconnection, by 3Opt and 4Opt moves.



a) b)

Figure 3. The crossings removal of two segments by a 2Opt move, may introduce new crossings. Case a) one crossing to be removed by 2Opt move; b) the appearance of two new crossings

It may be mentioned that these last two of moves may be effective even if the tour segments do not intersect.

```
for i=0 to n-3
    if ( d(city[i],city[i+1]) +
    d(city[i+2],city[i+3]) >
    d(city[i],city[i+2]) +
    d(city[i+1],city[i+3]) )
    {
          swap = city[i+1];
          city[i+1] = city[i+2];
          city[i+2] = swap;
    }
```

Figure 4. The algorithm of 2Opt moves

```
for i = 0 to n-6
    if( (d(city[i],city[i+1]) + d(city[i+2],city[i+3]) +
        d(city[i+4],city[i+5])) > (d(city[i],city[i+3]) +
        d(city[i+1],city[i+4] + d(city[i+2],city[i+5]) )
    {
          swap1 = city[i+1];
          city[i+1] = city[i+3];
          city[i+3] = swap1;
          swap1 = city[i+2];
          city[i+2] = city[i+4];
          city[i+4] = swap1;
    }
```

Figure 5. One type of the 3Opt moves

The algorithm of the 2Opt moves, seen in Figure 4, tests the distance length condition between the end points of the first and the third segments of the tour. If the condition holds then the end point of the first segment will be exchanged by the start point of the third segment. In the following, we check the condition between the end points of the second and the fourth segments of the tour, and so on. The tests continues until we reach the last segment of the tour i.e.($c_{n-1}c_n$). As the segment (c_nc_1) does not enter in this verification, the algorithm makes a shift of the indices in the cities tour with k positions. In this way, the next verification will enclose the former segment (c_nc_1). The same shift motion of the indices in the cities tour with k position is applied after each passing of the tour in the 3Opt and 4Opt algorithms of moves. Figure 5 shows an example of a 3Opt move, in which the exchange of the end points of segments does not change the order of the cities included in the partial paths between the checked segments. To guarantee the correctness of the tour, a routine which finds out and removes crossings between tour segments, was inserted, because we do not need to increase the running time by using to many kOpt (k = 2, 3, 4) moves,. This routine does not calculate the intersection point, but it is based on the cross product of two vectors i.e. tour segments (Cormen et al,2003).

3 NUMERICAL RESULTS

Several numerical tests were performed in order to establish the performances of the heuristic hybrid algorithm. The main source for the test instances was the TSPLIB database, available via anonymous ftp from softlib.rice.edu. The advantage of the problems from this source is that the optimum solution is already known.

The test problems were grouped in small instances where the number of the cities is between one to few hundreds, and medium to large instances, which have more than one thousand cities. Numerous runs were carryed out for each test instance on a Personal Computer, with an Intel Pentium processor at 1.6 GHz, 496 MB of RAM. The optimality ratio ρ, which expresses the quality of the results, was defined as:

$$\rho = 100 \cdot \left(\frac{S_{cost} - O_{cost}}{O_{cost}} \right) \%; \quad (1)$$

where:

- S_{cost} represents the solution cost,
- O_{cost} is the optimal cost.

Table 1. Optimality ratios for several test instances from TSPLIB

Test instance	Iteration	Optimality ratios ρ [%]
kroa100	21,285	0.01
krob100	22,197	0.25
kroc100	20,750	0.00
kroe100	22,165	0.43
kroa200	1,410,889	0.86
lin105	56,708	0.02
rd100	133,219	0.00
eil101	10,650,820	2.06
bier127	139,705,424	0.25
ch130	34,564,567	0.00
a280	5,951	4.76

rd400	139,896	3.25
rat575	3,556,944	5.52

In Table 1, 13 test instances and their optimality ratios in percent are listed, as well as the iteration at which these ratios were attained. In most of these test instances, the results reached exactly or almost the optimal value. The results are situated much under 1 percent above the optimal value. An exemption is the eil101 problem where, after 10,650,820 iterations, the result achieved has an optimality ratio of 2.06 percent higher than the optimal value. The test instances a280, rd400 and rat575 have a higher optimality ratio but, as we can see, the iteration number was not so high.

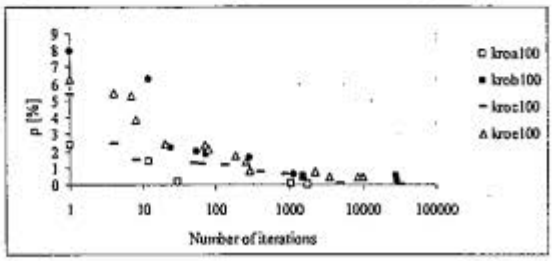


Figure 6. History of iterations for kroa100, krob100, kroc100 and kroe100

All of the run tests presented in Figure 6 and Figure 7 were completed according to the heuristic algorithm principles presented in Figure 1.

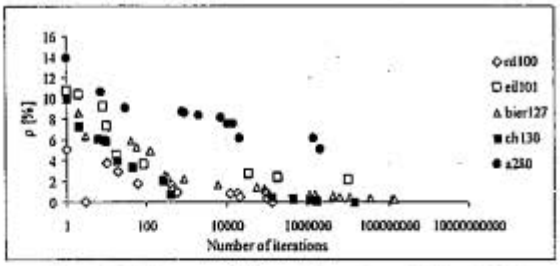


Figure 7. History of iterations for rd100, eil101, bier127, ch130 and a280

Consequently at each iteration, the tour construction procedure (i.e. global search) was followed by a local optimization (i.e. local search). To find out the progress made by the local optimization procedure, several tests were completed using the heuristic algorithm with tour construction procedure without and with local search optimization procedure respectively. Figure 8 shows the optimization progress in these two situations.

As we can see from Figure 8, local search optimization procedure improves the performance of the heuristic algorithm with few percent, especially at the beginning of the search process.

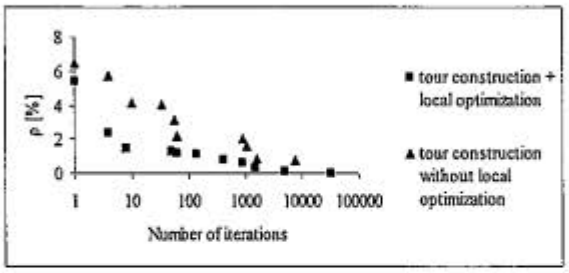


Figure 8. History of iterations for kroc100 without and with local optimization

There is almost always a gap of at least two to five percent between these two cases in the first one thousand iteration and the gap decreases as the number of iteration increases to a great extent. This is why the local optimization procedure plays an important role in the algorithm performance.

Table 2. Optimality ratios for large instances from TSPLIB

Test instance	Iteration	Optimality ratios ρ [%]
dsj1000	654,641	8.15
d1655	233,175	12.61
d2103	316,392	15.81
u1432	1,406,278	8.67
u2152	350,787	15.97

When the number of cities in the test instance exceeds one thousand, the search progress becomes quite slow. This is justifiable while the tour construction procedure is based on a random permutation approach, i.e. Monte Carlo method. As the number of the cities becomes higher, the number of trials in the random permutation approach must increase to a great extent to cover the interval of possible solution uniformly. It means that, depending on the type of selected convergence criterion of the algorithm, the iteration number or the running time have to increase accordingly.

,To determine the efficiency of the hybrid heuristic procedure, for kroa100 test problem, there were performed 50000 iterations. The results before and after the action of the local search procedure were saved at each iteration. In order to determine whether the two distributions differ significantly there were categorized the minimum path lengths, derived from both optimization methods, into nine categories. Each category covers a range of 500. It is evident from the Table 3, that there are very large discrepancies between corresponding counts.

Let O_i be the frequency of the i^{th} minimum path length category corresponding to the optimization method not using local search. Let E_i

be the frequency of the i^{th} minimum path length category corresponding to the optimization method using local search.

Table 3. The frequency table

	Without local search	With local search
Less than 21750	560	5248
21750 - 22250	5338	17356
22250 - 22750	12112	14086
22750 - 23250	13309	8412
23250 - 23750	10411	3786
23750 - 24250	5749	977
24250 - 24750	1984	126
24750 - 25250	454	8
More than 25250	83	1

Let E_i be the frequency of the i^{th} minimum path length category corresponding to the optimization method using local search. To measure the discrepancy between O_i and E_i we use the statistic X^2 given by:

$$X^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}; \qquad (2)$$

where k is the number of categories. The larger the value of X^2 the less the agreement between the frequencies O_i and E_i. The statistic X^2 has a chi-squared distribution with k-1 degrees of freedom.

The p-value is the criterion to determine whether the discrepancy between O_i and E_i is significant (see Table 4). If the p-value exceeds the 0.05 level of significance, we accept the null hypothesis that the two distributions of minimum path lengths are identical. Conversely, if the p-value is less than the 0.05 level of significance we deduce that the discrepancy between corresponding frequencies differ significantly, implying that the two distributions have contrasting shapes. For a large value of X^2 (78862.2), the p-value is expected to be very small, indicating that the two distributions are different.

Table 4. Test statistics

	Category
Chi-Square	78862.179
df	8
P-value	0.000

Moreover, statistical measures for central tendency, dispersion, peakedness and symmetry for the two distributions are very contrasting. By comparing the mean, standard deviation, kurtosis and skewness of the two distributions (see Table 5), one notices that the two-step optimization procedure is yielding a higher proportion of minimum path lengths that are closer to the optimal solution (21285) implying that it is more efficient than the one step-procedure.

Table 5. Statistics

	Without local search	With local search
N	50000	50000
Mean	23056.3471	22414.1705
Median	23006.8603	22322.1387
Std. Deviation	685.75857	583.48231
Skewness	.363	.638
Kurtosis	-.189	.037
Minimum	21322.74	21285.44
Maximum	26088.29	25329.03

This fact is very well shown in Figure 9 by the the distribution of the minimum path length, for kroa100 test problem, displayed on a normal curve.

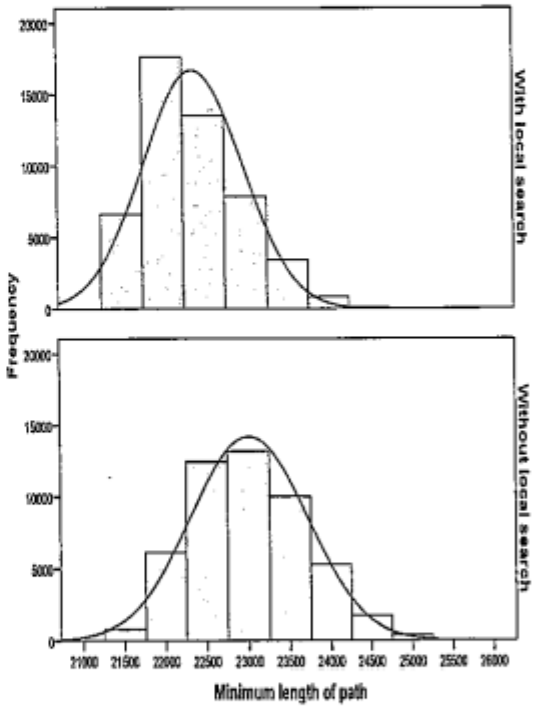


Figure 9. The distribution of the minimum path length, for kroa100 test problem

It can be easily seen how much influence the local search procedure has on the distribution of the iterative search results, that tend to move towards the optimum direction. The more efficient the local search procedure is, the larger the difference between the two distributions in the Figure 9 will be.

4 CONCLUSION

In this paper was presented a hybrid heuristic for solving the Traveling Salesman Problem,

designed on the frame of a general optimization procedure which act upon two steps, iteratively. In the first step of the global search, a feasible tour is constructed based on insertion approach. In the second step the feasible tour found at the first step, is improved by a local search optimization procedure.

One of the hard problems of the algorithm design was not only to generate a random permutation, but to avoid the repetition of the same permutation sometime, along the iterative steps of the search process. This thing is more complicated to verify as the number of the cities becomes higher. Because there is a strong connection between the probability of escape from a local optimum point and the random permutation sequence in which the cities are inserted in tour, a main future research consideration will be the improvement of the procedure which generates the random permutation of the cities. The statistical analysis of the kroa100 test problem shows the effectiveness of the local search optimization procedure, in the graphical representation.

A key point in the future research will be the improvement of the local search procedure. For the moment, only kOpt moves were used as a local search approach. Now, the results are close to those of the similar heuristic techniques from the literature. An intensive search in the neighbourhood of good solutions by making some perturbations, similar to the Simulated Annealing or the use of the search history to determine the so called backbones (Schneider,2002), (Schneider,2003), might be a possible approach in the improvement of the local search that will follow.

5 REFERENCES

►Angéniol, B. et al, (1988) Self-organizing feature maps and the traveling salesman problem. Neural Networks 1, p.289-293

►Applegate, D., Cook, W. (1993) Solving large-scale matching problems, in: D. Johnson, C.C. McGeoch (Eds.) Network Flows and Matching, DIMACS Series in Discrete Mathematics and Theoretical Society, vol. 12, p. 557-576

►Bentley, J.L. (1992) Fast algorithms for geometric traveling salesman problem. ORSA J. Comput. 4, p.387-411

►Cormen, T.H. et al. (2003) Introduction to algorithms. Second ed., Cambridge, Massachusetts: The MIT Press

►Croes, G.A. (1958) A method for solving traveling salesman problems. Operations Research 6, p.791-812

►Crowder, H., Padberg, M. (1980) Solving large-scale symmetric travelling salesman problems to optimality. Management Science 26, p.495-509

►Dueck, G., Scheuer, T. (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J. Comput. Phys. 9, p.161-175

►Frieze, A.M. (1979) Worst-case analysis of algorithms for traveling salesman problems. Methods of Operations Research 32, p.97-112

►Glover, F. (1986) Future paths for integer programming and links to the artificial intelligence. Computers and Ops.Res. 13, p.533-549

►Goertzel, B. (1993) The structure of intelligence, Berlin: Springer-Verlag

►Grötschel, M., Holland, O. (1991) Solution of Large-scale Symmetric Travelling Salesman Problems. Mathematical Programming 51, p.141-202

►Johnson, D.S., McGeoch, L.A. (1997) The Traveling Salesman Problem: A Case Study in Local Optimization, in: E.H.L. Aarts, & J.K. Lenstra, Local Search in Combinatorial Optimization, John Wiley and Sons, London, p. 215-310

►Kirkpatrick, S. et al. (1983) Optimization by simulated annealing. Science 20, p.671-680

►Knox, J. (1994) Tabu search performance on the symmetric traveling salesman problem. Computers and Ops.Res. 21, p.867-876

►Kohonen, T. (1988) Self-Organization and Associative Memory, Springer-Verlag, Berlin

►Lawler, E.L. et al. (1985) The Traveling Salesman Problem. Chichester: John Wiley and Sons

►Lin, S. Kernighan, B.W. (1973) An effective Heuristic Algorithm for the Traveling Salesman Problem. Operations Res. 21, p.498-516

►Metropolis, N. et al. (1953) Equation of state calculation for fast computing machines. J. Chem. Phys. 21, p.1087-1092

►Moscato, P., Fontanari, J.F. (1990) Stochastic vs. deterministic update in simulated annealing. Phys.Lett. A 146, p.204-208

►Padberg, M.W., Rinaldi, G. (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problems. SIAM Review, 33, p.60-100

►Padberg, M.W., Rinaldi, G. (1987) Optimization of a 532 city Symmetric Traveling Salesman Problem by Branch and Cut. Operations research Letters 6, p.1-7

►Peterson, C. (1990) Parallel distributed approaches to combinatorial optimization:

Benchmark studies on traveling salesman problem. Neural Computation 2, p.261-269

►Peterson, C., Söderberg, B. (1989) A new method for mapping optimization problems onto neural network. International Journal of Neural Systems 1, p.3-22

►Press, W.H. et al. (1997) Numerical Recipes in C. The Art of Scientific Computing, 2nd ed., Cambridge University Press, New York

►Rosenkrantz, D.J. et al. (1977) Analysis of several heuristics for the traveling salesman problem. SIAM J. Comput., 6, p.563-581

►Schneider, J. (2002) The time-dependent traveling salesman problem. Physica A, 314, p.151-155

►Schneider, J. (2003) Searching for backbones - a high performance parallel algorithm for solving combinatorial optimization problems. Future Generation Computer Systems 19, p.121-131

►Simmen, M.W. (1991) Parameter sensitivity on the elastic net approach to the traveling salesman problem. Neural Computation 3, p.363-374.

►Ugajin, R. (2002) Method to solve the traveling salesman problem using the inverse of diffusion process. Physica A, 307, p.260-268.

►Valenzuela, C.L., Jones, A.J. (1994) Evolutionary divide and conquer (I): A novel genetic approach to the TSP. Evolutionary Computation 1, p.313-333.

►Xu, X., Tsai, T. (1991) Effective neural algorithms for traveling salesman problem. Neural Networks 4, p.193-205.