# 5

# A Fast Evolutionary Algorithm for Traveling Salesman Problem

Xuesong Yan, Qinghua Wu and Hui Li
*School of Computer Science, China University of Geosciences,*
*Faculty of Computer Science and Engineering, Wu-Han Institute of Technology, China*

## 1. Introduction

The traveling salesman problem (TSP)[1] is one of the most widely studied NP-hard combinatorial optimization problems. Its statement is deceptively simple, and yet it remains one of the most challenging problems in Operational Research. The simple description of TSP is: Give a shortest path that covers all cities along. Let G = (V; E) be a graph where V is a set of vertices and E is a set of edges. Let C = (cij) be a distance (or cost) matrix associated with E. The TSP requires determination of a minimum distance circuit (Hamiltonian circuit or cycle) passing through each vertex once and only once. C is said to satisfy the triangle inequality if and only if cij + cjk ≧ cik for all i, j, k  V.

Due to its simple description and wide application in real practice such as Path Problem, Routing Problem and Distribution Problem, it has attracted researchers of various domains to work for its better solutions. Those traditional algorithms such as Cupidity Algorithm, Dynamic Programming Algorithm, are all facing the same obstacle, which is when the problem scale N reaches to a certain degree, the so-called "Combination Explosion" will occur. For example, if N=50, then it will take $5\times10^{48}$ years under a super mainframe executing 100 million instructions per second to reach its approximate best solution.

A lot of algorithms have been proposed to solve TSP[2-7]. Some of them (based on dynamic programming or branch and bound methods) provide the global optimum solution. Other algorithms are heuristic ones, which are much faster, but they do not guarantee the optimal solutions. There are well known algorithms based on 2-opt or 3-opt change operators, Lin-Kerninghan algorithm (variable change) as well algorithms based on greedy principles (nearest neighbor, spanning tree, etc). The TSP was also approached by various modern heuristic methods, like simulated annealing, evolutionary algorithms and tabu search, even neural networks.

In this paper, we proposed a new algorithm based on Inver-over operator, for combinatorial optimization problems. In the new algorithm we use some new strategies including selection operator, replace operator and some new control strategy, which have been proved to be very efficient to accelerate the converge speed. At the same time, we also use this approach to solve dynamic TSP. The algorithm to solve the dynamic TSP problem, which is the hybrid of EN and Inver-Over algorithm.

## 2. Tradition approaches for travel salesman problems

### 2.1 Genetic algorithm (GA)

Genetic Algorithm is based on the idea of Darwin evolutionism and Mendel genetics that simulates the process of nature to solve complex searching problems. It adopts the strategy of encoding the population and the genetic operations, so as to direct the individuals' heuristic study and searching direction. Since GA owns the traits of self-organization, self-adaptation, self-study etc, it breaks away from the restriction of the searching space and some other auxiliary information. However, when facing different concrete problems (e.g. TSP), it's always necessary for us to seek better genetic operators and more efficient control strategy due to the gigantic solution space and limitation of computation capacity.

Use GA solve the TSP, including the following steps:

Chromosome Coding: In this paper, we will use the most direct way to denote TSP-path presentation. For example, path 4-2-1-3-4 can be denoted as (4, 2, 1, 3) or (2, 3, 1, 4) and it is referred as a chromosome. Every chromosome is regarded as a validate path. (In this paper, all paths should be considered as a ring, or closed path).

Fitness value: The only standard of judging whether an individual is "good" or not. We take the reciprocal of the length of each path as the fitness function. Length the shorter, fitness values the better. The fitness function is defined as following formula:

$$f(Si) = \Big/ \sum_{i=1}^{N} d[C_{n(i)}, C_{n(i+1) \bmod N}] \qquad (1)$$

Evolutionary Operator or Evolutionary Strategy: they are including selection operator, crossover operator and mutation operator.

Selection Strategy: After crossover or mutation, new generations are produced. In order to keep the total number consistent, those individuals who are not adjust to the environment must be deleted, whereas the more adaptive ones are kept.

Probability of mutation: In order to keep the population sample various, and prevent from trapping into local minimum, mutation is quite necessary. But to seek higher executing speed, the probability of mutation must be selected properly.

Terminal conditions: The stop condition of the algorithm.

### 2.2 A fast evolutionary algorithm based on inver-over operator

Inver-over operator has proved to be a high efficient Genetic Algorithm[2]. The creativity of this operator is that it adopts the operation of inversion in genetic operators, which can effectively broaden the variety of population and prevent from local minimum and lead to find the best solutions quickly and accurately. GT algorithm[2] is be perceived as a set of parallel hill-climbing procedures.

In this section, we introduce some new genetic operators based on GT. And there are also some modifications on some details, which aim to quicken the convergence speed. Numerical experiments show the algorithm can effectively keep the variety of population, and avoid prematurely in traditional algorithms. It has proved to be very efficient to solve small and moderate scale TSP; particularly, larger-scale problem can also get a fast convergence speed if we choose a small population that involves the evolution.

Selection Operator: Randomly select two chromosomes $S1, S2$ from population {P}, let f ($S2$)>f ($S1$); randomly select a gene segment $\Delta s1$ from $S1$, then judge if there is gene segment $\Delta s2$ in $S2$ that meets the conditions below: it has the same length (number of cities) and the starting city with $\Delta s1$. If $\Delta s2$ exists, replace $\Delta s1$ by $\Delta s2$ in chromosome $S1$, then the rest genes are adjusted follow the partly mapping rules.

Replace Operator: Randomly select two chromosomes $S1, S2$ from population {P}, let f ($S2$)>f ($S1$), let $S1$ be the parent, then randomly select a gene segment $\Delta s1$ in $S1$, then judge if there is a gene segment $\Delta s2$ exists in $S2$ that meets the conditions below: it has the same length (number of cities) and cities with $\Delta s1$, only the sequence of cities varies. If it exists, judge the distance of the two segments $\Delta s1$, $\Delta s2$. If $\Delta s2$ is shorter, replace $\Delta s1$ by $\Delta s2$ in $S1$; else quit.

The new algorithm mainly involves crossover operator, but crossover relies strongly on the current population and may be easily trapped into local minimum. So mutation is introduced and can effectively solve the problem. But mutation itself is blind; it can act efficiently at the beginning of the algorithm, but when it's converged round to the approximate best solution, the probability of successfully optimize the chromosome turns out to be very low. So Dynamically alter the probability of selecting mutation operator is necessary. We use the formula below to alter the probability of selecting mutation operator:

$$p = p \times (1 - GenNum \times 0.01 / \max Gen) \tag{2}$$

$p$ is the probability of mutation, GenNum is the current evolutionary times; maxGen is the max evolutionary times when algorithm stops. Fig.2 provides a more detailed description of the whole algorithm in general.

Random initialization of the population $P$
While (not satisfied termination condition) do
i=0
Begin
for each individual $S_i \in P$ do
begin (if i!=N)
{ $S' \leftarrow S_i$
Select (randomly) a City $C$ from $S'$
repeat
{begin
if ($rand() \le p_1$)
{select the city $C^{'}$ from $S'$
invert the gene segment between $C$ and $C^{'}$
}
else
{select (randomly) an individual $S^{''}$ in $P$
assign to $C^{'}$ the next city to the city $C$ in the select individual
}
if (the next city or the previous city of city $C$ in $S'$ is $C'$)
exit repeat loop

else
{invert the gene segment between $C$ and $C'$
calculate d
if(d<0 and evolutionary speed< critical speed)
exit repeat loop
}
$C \leftarrow C'$
end
}
if $(eval(S') \le eval(S_i))$
$S_i \leftarrow S'$
i = i +1}
end
calculate evolutionary speed and update the probability of mutation
if (evolutionary speed< critical speed and $(rand() \le p_2)$
{select (randomly) $S_1$ , $S_2$ from $P$
select (randomly) gene segment $\Delta s1$ from $S_1$
if ( $\Delta s1$ = $\Delta s2$ and the starting city is same) // $\Delta s2$ in $S_2$
replace $\Delta s1$ with $\Delta s2$
}
Select the better gene segments from those adaptive chromosomes
if $(eval(g_1) \le eval(g_2))$
$g_1 \leftarrow g_2$
end

In this section we present the experimental results of the proposed algorithm. All experiments are performed on PⅣ 2.4GHz/256M RAM PC. In the experiments all test cases were chosen from TSPLIB (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95). The optimal solution of each test case is known. The parameters for algorithm are as Table.1.

| | |
|---|---|
| Size of Population | 100 |
| Mutation Probability | 0.02 |
| Selection Probability | 0.05 |
| Critical Speed | 5000 |

Table 1. Parameters for the algorithm

We list the test cases and their optimal solutions in Fig.1.
The above results demonstrate clearly the efficiency of the algorithm. Note that for the seven test cases the optimum was found in all ten runs. The number of cities in these test cases varies from 70 to 280. Note also, that the running time of the algorithm was reasonable: below 3 seconds for problems with up to 100 cities, below 10 seconds for the test case of 144 cities, below 40 seconds for the test case with 280 cities.

| Instance | Result in TSPLIB | Optimum in TSPLIB | Our Result | Time |
|---|---|---|---|---|
| st70 | 675 | 678.597 | 677.109 | 0.67 |
| eil76 | 538 | 545.387 | 544.369 | 1.16 |
| kroA100 | 21282 | 21285.443 | 21285.443 | 1.69 |
| rd100 | 7910 | 7910.396 | 7910.396 | 2.14 |
| Pr136 | 96772 | 96772 | 96770.924 | 7.11 |
| Pr144 | 58537 | 58537 | 58535.221 | 7.97 |
| a280 | 2579 | 2856.769 | 2856.769 | 33.47 |

Fig. 1. Results of the algorithm

## 3. Information dynamic traveling salesman problem

Most research in evolutionary computation focused on optimization of static, no-change problems. Many real world optimization problems however are actually dynamic, and optimization methods capable of continuously adapting the solution to a changing environment are needed. The main problem with standard evolutionary algorithms used for dynamic optimization problems appears to be that EAs eventually converge to an optimum and thereby loose their diversity necessary for efficiently exploring the search space and consequently also their ability to adapt to a change in the environment when such a change occurs. Since Psaraftis [11] first introduced DTSP, some research works have touched on this area [3,4, 6,7,9,10]. However the research is just at the initial phase and quite a few crucial questions.

### 3.1 The elastic net method to TSP

The elastic net was first proposed as a biologically motivated method for solving combinatorial optimization problems such as the traveling salesman problem (TSP) [1]. In the method, let the coordinates of a city, i, be denoted by the vector ci, and let the coordinates of the route point (or unit), u, be tu. Define the TSP in terms of a mapping from a circle to a plane. The TSP consists of finding a mapping from a circle, h, to a finite set of points, C=ci…cN, in the plane, which minimizes the distance:

$$D = \oint_h f(s)ds \tag{3}$$

where the circle, h, is parameterized by arc-length, s, and f(s) provides a unique mapping from each point on h to a point on the plane. The locus of points defined by f(s) forms a loop, l, in the plane which passes through each city exactly once.

The loop, l, is modeled as finite set, t1..tM, of points, or units. Due to the difficulty in obtaining a one-to-one mapping of units to plane points (cities) it is customary to set M > N in order that each city can become associated with at least one unit.

The energy function to be minimized is:

$$E = -\alpha \sum_{i=1}^{N} \ln \sum_{u=1}^{M} \Phi(|c_i - t_u|, K) - k \sum_{u=1}^{M} \ln \sum_{i=1}^{N} \Phi(|c_i - t_u|, k) + \beta \sum_{u=1}^{M} (|t_u - t_{u+1}|)^2 \tag{4}$$

where $\varphi$(d,K)=exp(-(d2/2K2)), and K is the standard deviation of the function, $\varphi$.
Differentiating E with respect to tu, and multiplying through by K, gives the change $(=-K\partial E/\partial t_u)$, $\triangle t_u$, in the position of a unit, u:

$$\Delta t_u = (\alpha \sum_{i=1}^{N} f_{ui} + k \sum_{u=1}^{M} b_{ui})(c_i - t_i) + \beta K(t_{u+1} - 2t_u + t_{u-1}) \tag{5}$$

where fui is like follow:

$$f_{ui} = \frac{\Phi(|c_i - t_u|, K)}{\sum_{v=1}^{M} \Phi(|c_i - t_v|, K)} \tag{6}$$

This term ensures that each city become associated with at least one unit. It is the Gaussian weighted distance of tu from ci expressed as a proportion of the distances to all other units from ci. In contrast, the term bui ensures that each unit becomes associated with at least one unit. It is the Gaussian weighted distance of ci from tu, expressed as proportion of the distances to all other cities from tu:

$$b_{ui} = \frac{\Phi(|c_i - t_u|, K)}{\sum_{v=1}^{M} \Phi(|c_i - t_v|, K)} \tag{7}$$

The EN method [8] is analogous to laying a circular loop of elastic on a plane containing a number of points (cities) and allowing each city to exert an attractive force on the loop. (In practice the EN method models this loop as finite set of points). Each city is associated with a Gaussian envelope which (using our physical analogy) effectively means that each city sits in a Gaussian depression on the plane. The standard deviation of the Gaussians associated with all cities is identical. Each city attracts a region on the elastic loop according to the proximity of that region to a given city and the standard deviation of the Gaussian. Initially the standard deviation is large, so that all regions on the loop are attracted to every city to the same extent (approximately) as every other region. As the standard deviation is decreased a city differentially attracts one or more nearby regions on the loop, at the expense of more distant loop regions. These loop-city interactions are modulated by the elasticity of the loop, which tends to keep the loop length short; the elasticity also tends to ensure that each city becomes associated with only one loop region. For a given standard deviation the integral of loop motion associated with each city is constant, and is the same for all cities, so that each city induces the same amount of motion of the loop. By slowly decreasing the standard deviation each city becomes associated with a single point on the loop_ Thus the loop ultimately defines a route of the cities it passes through. If the standard deviation is reduced sufficiently slowly then the loop passes through every city; and therefore defines a complete route through all cities.

### 3.2 The elastic net method to dynamic TSP

Dynamic TSP (D-TSP)[4] is a TSP determined by the dynamic cost (distance) matrix as follows:

$$D(t) = \{d_{ij}(t)\}_{n(t)\times n(t)} \tag{8}$$

where dij(t) is the cost from city(node) i to city j, t is the real world time. This means that the number of cities n(t) and the cost matrix are time dependent that's (1)some cities may appear, (2)some may disappear and (3)the locations of some may be modified with time going on. These are the three types of actions to a D-TSP.
The D-TSP solver should be designed as solutions of a two-objective optimization problem. One is to minimize the size of time windows:

$$s_k = t - t_k \tag{9}$$

where t tk. The other is to minimize the length of $\pi(t_k) = (\pi_1, \pi_2, ..., \pi_{n(t_k)})$ :

$$d(\pi(t_k)) = \sum_{i=1}^{n(t_k)} d_{\pi_i, \pi_{i+1}}(t_k) \tag{10}$$

where $\pi_{n(t_k)+1} = \pi_1$ .

The two objectives mean that the solver is to find the best tour in the minimal time window. If this aim cannot be satisfied, we can make some tradeoffs between the two objectives.
We introduce the thought of the EN method into dynamic TSP. First, suppose we have already attained the maximum optimum of static TSP. Second, add a dynamic point. When the dynamic point moves to a position, we can use the EN method to attain the maximum optimum route. The whole processing like Fig.2 shows.

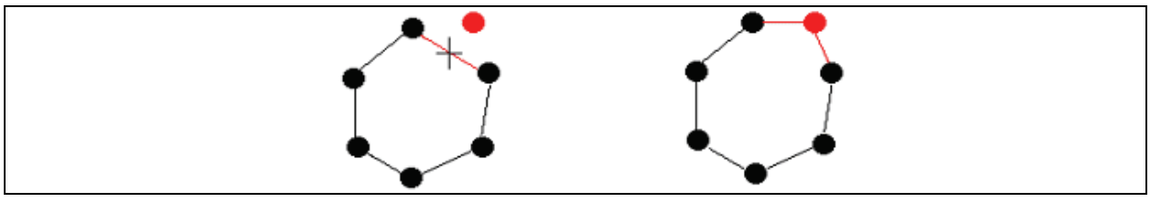


Fig. 2. The processing of EN method to dynamic point

Because the dynamic TSP needs give the maximum optimum in real-time, so the computation speed very high. Gou's algorithm [2] with Inver-Over operator is a very efficient evolutionary algorithm for static TSP with high speed and more details about this algorithm can be got in [5]. We convert it with EN method to solve dynamic TSP. The algorithm code like following shows.

```
DEN Algorithm ()
{
  Initiate parameter;
  Initiate population {Pi};
  While true do
  {
    Attain dynamic point position (x', y');
    Inver-Over (x [], y [], path [], M);
    EN (x [], y [], path [], x', y');
  }
}
```

### 3.3 Experiments and results

We have evaluated the algorithm and test it with the data set from TSPLIB (http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95). Fig.3 is the example of the DEN algorithm to KeroA150 in different time. In the example, the red point is the dynamic point and it moves around circle. From the example, we can see in different time, the dynamic point in different position and has the different distance. It shows this example is a dynamic problem. Through the experiment, we find the DEN algorithm cannot guarantee the route to be the maximum optimum, but this algorithm can move the dynamic point and confirm it to the new route very fast. So, it is very valid to solve dynamic TSP problem.

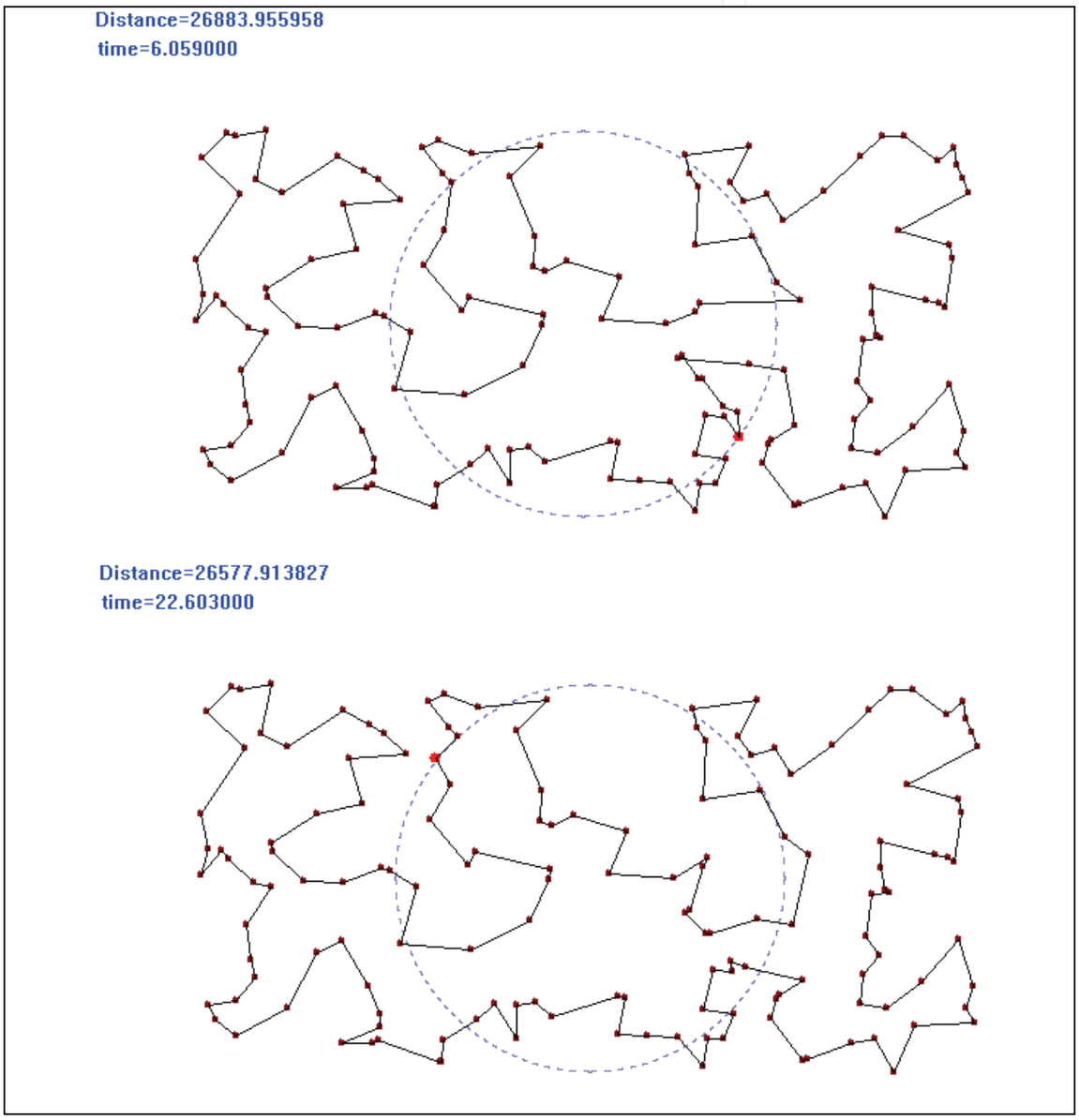



Fig. 3. DEN algorithm to KeroA150 in different time

## 4. Conclusion

In this paper, we introduce a fast evolutionary algorithm for combinatorial optimization problem. This algorithm is based on Inver-over operator, in the algorithm we use some new strategies including selection operator, replace operator and some new control strategy, which have been proved to be very efficient to accelerate the converge speed. The new algorithm shows great efficiency in solving TSP with the problem scale under 300. Particularly, if we choose a comparatively smaller population scale that involves evolution, the algorithm is also efficient to get the approximate best solution in a short executing time.
In this paper, we also introduce an approach to solve dynamic TSP-- Elastic Net Method. A dynamic TSP is harder than a general TSP, which is a NP-hard problem, because the city number and the cost matrix of a dynamic TSP are time varying. And needs high computation speed. We propose the algorithm to solve the dynamic TSP, which is a hybrid of EN method and Inver-Over. Through the experiment, we got good results. Some strategies may increase the searching speed of evolutionary algorithms for dynamic TSP, forecasting the change pattern of the cities and per-optimizing, doing more experiments including changing the city number, and etc. These will be our future work.

## 5. References

Durbin R, Willshaw D. (1987). An Anlaogue Approach to the Traveling Salesman Problem Using an Elastic Net Approach. Nature, 326, 6114, pp. 689-691.

T.Guo and Z.Michalewize. (1998). Inver-Over operator for the TSP. In Parallel Problem Sovling from Nature(PPSN V),Springer-Verlag press, pp. 803-812.

Z.C.Huang, X.L.Hu and S.D.Chen. (2001). Dynamic Traveling Salesman Problem based on Evolutionary Computation. In Congress on Evolutionary Computation(CEC'01), pp. 1283-1288, IEEE Press.

Aimin Zhou, Lishan Kang and Zhenyu Yan. (2003). Solving Dynamic TSP with Evolutionary Approach in Real Time. In Congress on Evolutionary Computation(CEC'03), pp. 951-957, IEEE Press.

H.Yang, L.S.Kang and Y.P.Chen. (2003). A Gene-pool Based Genetic Algorithm for TSP. Wuhan University Journal of Natural Sciences 8(1B), pp. 217-223.

X.S.Yan, L.S.Kang et.al: (2004). An Approach to Dynamic Traveling Salesman Problem. Proceedings of the Third International Conference on Machine Learning and Cybernetics, pp. 2418-2420, IEEE Press, Shanghai.

X.S.Yan, A.M Zhou, L.S Kang et,al; (2004). TSP Problem Based on Dynamic Environment. Proceedings of the 5th World Congress on Intelligent Control and Automation, pp. 2271-2274, IEEE press, Hangzhou, China.

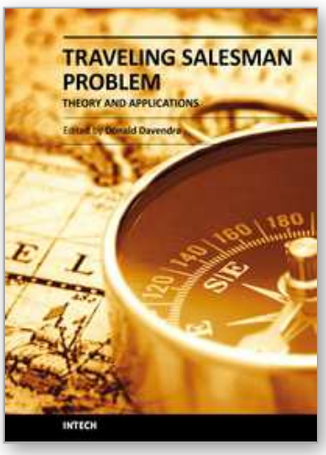
J.V.Stone. (1992). The Optimal Elastic Net: Finding Solutions to the Traveling Salesman Problem. In Proceedings of the International Conference on Artificial Neural Networks, pp. 170-174, Brighton, England.

J.Branke. (2002). Evolutionary Optimization in Dymnamic Environments. Kluwer Academic Publishers.

C.J.Eyckelhof and M.Snoek. (2002). Ant Systems for a Dynamic TSP -Ants caught in a traffic jam. In 3rd International Workshop on Ant Algorithms (ANTS2002), pp. 88-99, Springer Press, Brussels, Belgium.

H.N.Psaraftis. (1998). Dynamic vehicle routing problems. In Vehicle Routing: Methods and Studies, B.L.Golden and A.A.Assad(eds), pp. 223-248, Elsevier Science Publishers.

X.S.Yan, Hui Li et.al: (2005). A Fast Evolutionary Algorithm for Combinatorial Optimization Problems. Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, pp. 3288-3292, IEEE Press.




**Traveling Salesman Problem, Theory and Applications**
Edited by Prof. Donald Davendra

ISBN 978-953-307-426-9
Hard cover, 298 pages
**Publisher** InTech
**Published online** 30, November, 2010
**Published in print edition** November, 2010

This book is a collection of current research in the application of evolutionary algorithms and other optimal algorithms to solving the TSP problem. It brings together researchers with applications in Artificial Immune Systems, Genetic Algorithms, Neural Networks and Differential Evolution Algorithm. Hybrid systems, like Fuzzy Maps, Chaotic Maps and Parallelized TSP are also presented. Most importantly, this book presents both theoretical as well as practical applications of TSP, which will be a vital tool for researchers and graduate entry students in the field of applied Mathematics, Computing Science and Engineering.



**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen