Simple learning algorithm for the traveling salesman problem

Kan Chen
Department of Computational Science, National University of Singapore, Singapore 119260

(Received 7 October 1996; revised manuscript received 3 December 1996)

We propose a learning algorithm for solving the traveling salesman problem based on a simple strategy of trial and adaptation. (i) A tour is selected by choosing cities probabilistically according to the ''synaptic'' strengths between cities. ii) The ''synaptic'' strengths of the links that form the tour are then enhanced (reduced) if the tour length is shorter (longer) than the average result of the previous trials. We perform extensive simulations of the random distance traveling salesman problem. For sufficiently slow learning rates, near-optimal tours can be obtained with the average tour lengths close to the lower bounds for the shortest tour lengths. [S1063-651X(97)12603-4]

PACS number(s): 07.05.Mh, 89.80.+h, 02.60.Pn

Many problems in science and engineering can be formulated in terms of optimization problems. In physics, examples of optimization include predicting the most stable configuration of molecules and finding the ground state configuration of spin glasses. Optimization problems are usually easy to formulate but hard to solve. Particularly hard are a class of interesting optimization problems that are NP complete: an exact solution requires a number of computational steps that grows exponentially with the size of the problem. The traveling salesman problem (TSP), which consists of finding the shortest closed tour connecting all cities in a map, is a classic example and a good testing ground for optimization methods. Because exact solutions are almost impossible to obtain the aim is to find near-optimal solutions.

A few optimization methods, based on ideas from physics and biology, have been developed recently which lead to rather good *general purpose* algorithms [1] for solving optimization problems. They have been successfully applied to a wide range of practical problems. One of them is a stochastic algorithm known as optimization by simulated annealing (OSA) [2]. The algorithm consists of an evolution according to Monte Carlo dynamics performed at a sequence of effective temperatures to simulate the annealing effect. The stochastic dynamics allows access to a larger region of configuration space than simple ''quenching'' methods, and helps in reaching a good solution. However, the Monte Carlo search is based on the evolution of a *single* configuration, thus is often confined to a limited region of configuration space and is not efficient in searching through the configuration space. For this reason the performance of OSA is quite sensitive to the choice of an annealing schedule and the initial configuration. Another general purpose algorithm which has been used extensively is the genetic algorithm (GA) [4], which has also been applied to the traveling salesman problem [5,6]. The genetic algorithm demonstrates the importance of keeping many configurations (''species'') in the optimization process. The algorithm mimics the evolutionary tools of reproduction, mating, and mutation. It has been shown that the algorithm performs well for small-size TSP's [5]. For large-size problems, however, it is not clear that current genetic algorithms are efficient because one has to keep a significant number of configurations. There are many variants of these two general algorithms using different local search methods. For example, the annealing algorithm can be improved using the multicanonical method; this has also been applied to TSP [3]. For the general performance of such algorithms, the reader is referred to the recent review article by Johnson and McGeoch [7].

In this paper, we propose a simple learning algorithm which appears to have the advantages of both the OSA and genetic algorithms. The preliminary version of our algorithm was presented in Ref. [9]. Our algorithm may be viewed as a neural network algorithm, but differs completely from Hopfield and Tank's approach [8], which is equivalent to gradient descent of an energy function with the tour length as a major term. In our approach the cities can be viewed as ''neurons'' and the connections between them as ''synapses.'' Initially a synaptic strength w_{ij} is assigned to each pair of cities $\{ij\}$:

$$w_{ij}=e^{-d_{ij}/T}, \tag{1}$$

where d_{ij} is the distance between them and T is a parameter which controls the relative initial strengths of synapses. The basic ingredients in the algorithm are *trial and adaptation*: first select a tour; then modify the synaptic strengths of the links which formed the tour to favor tours with shorter tour lengths by comparison. This procedure is then repeated.

Selection of tour. A tour is selected by picking cities sequentially with the probability determined by the synaptic strengths. A simple selection procedure consists of the following steps. The first city i_1 is picked at random, the second one i_2 is picked with probability $P_{i_2i_1}\propto w_{i_2i_1}$, then the third one i_3 is picked among available cities with probability $P_{i_3i_2}\propto w_{i_3i_2}$, and so on. For computational efficiency, however, we set up a priority list of neighbors for each city and use the priority lists to generate the tour. The first two positions in the priority list are selected probabilistically with probabilities proportional to the corresponding synaptic strengths with the neighbors, and the remaining positions in the list are ordered according to the synaptic strengths of the remaining neighbors. Since the synaptic strengths change slowly during the simulation, the new priority lists can be set up quickly before each trial using the previous lists. The tour is now generated as follows. The first city i_1 is again picked at random, the second city i_2 chosen as the first available city in the priority list of i_1, the third city i_3 is chosen as the first available city in the priority list of i_2, and so on.

The tour obtained is then improved using the exhaustive search for two-bond rearrangements [10]. This generates the tour of the current trial, which will be used for comparison with the tours obtained from the previous trials. The search for two-bond rearrangements allows tours of better quality, particularly in the beginning of the simulation process, to be used in tour evaluation and modification of synaptic strengths. This helps to improve the learning algorithm significantly. Note that with the use of the priority lists the exhaustive search can be made efficient, because only the links with low priority need to be swapped. As the learning process converges, the configurations picked using the priority lists are already near-optimal, and very few two-bond rearrangements are needed to improve them.

Modification of synaptic strengths. The tour obtained in the current trial is compared with the tours obtained in the previous trials. In our simulation we keep a number of tours (denoted by m below) for the purpose of comparison. Let $\{i_1 i_2 \cdots i_N\}$ denote the current tour and $\{i'_1 i'_2 \cdots i'_N\}$ denote one of the previous tours. Let the tour lengths for these two tours be d and d', respectively. The comparison of these two tours leads to the following modification of the synaptic strengths of the links that form the tours:

$$w^{\mathrm{new}}_{i_l i_{l+1}} = w^{\mathrm{old}}_{i_l i_{l+1}} e^{-\frac{\alpha}{m}(d-d')}, \quad l=1,\ldots,N,$$

$$w^{\mathrm{new}}_{i'_l i'_{l+1}} = w^{\mathrm{old}}_{i'_l i'_{l+1}} e^{-\frac{\alpha}{m}(d'-d)}, \quad l=1,\ldots,N, \qquad (2)$$

where $i_{N+1}=i_1$ and $i'_{N+1}=i'_1$. α represents the modification rate of synaptic strengths (since m comparisons are made at each stage, the coefficient in the exponent is written as α/m). According to this rule, if the current tour length is shorter (longer) than the previous one, the corresponding synaptic strengths for the links that form the tour are enhanced (reduced). Another tour selection is made acording to the prescription given earlier and the procedure repeated. As learning advances, some links are gradually abandoned, while others are increasingly favored: eventually the selection converges to a near-optimal tour.

In this paper, we study the random distance TSP, in which the distances between cities are independent random variables between 0 and 1. Part of the reason we choose the random distance TSP is that many algorithms based on simple local search methods give very poor results for these instances. Besides, we can also compare our results with the extensive numerical studies of Krauth and Mézard [11], who also obtained exactly the length (≈ 2.0415) of the optimal tour in the limit of a large number of cities for this problem.

Let us first discuss the choice of the parameters used in our simulation. The parameter T [see Eq. (1)] is chosen to be of the order of the shortest distance so that the synaptic strength for the long distance link is made very small initially. In our simulation we simply choose $T=1/N$, where N is the number of cities. The most important parameter in our simulation is α, which controls the rate of learning. The choice of the parameter m is not crucial. In our simulations we simply fix $m=50$ (i.e., we keep the 50 most recent tours).

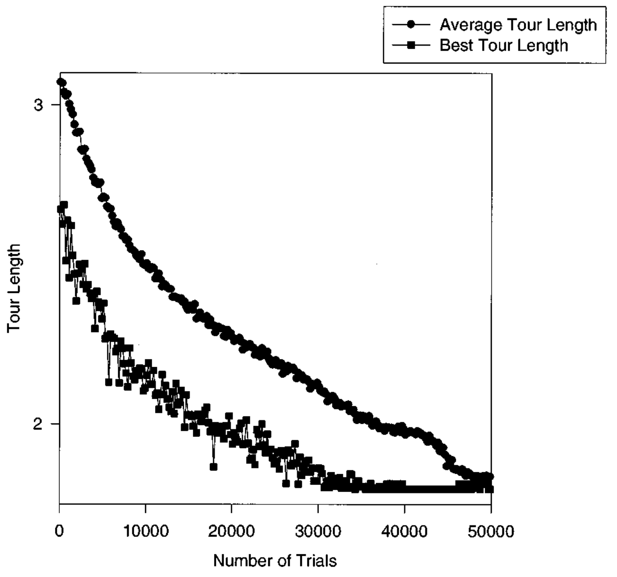

FIG. 1. Tour length vs number of trials for the optimization of a 200-city random distance TSP: The upper curve shows the average tour lengths calculated in consecutive intervals of 200 trials; the lower curve shows the shortest tour lengths in these intervals. Data are taken every 200 trials, and the optimization is performed with $\alpha=0.03$.

The general performance of our algorithm is illustrated in our study of a 200-city TSP. The result of the simulation with $\alpha=0.03$ is plotted in Fig. 1, which shows both the average tour lengths and the shortest tour length in successive intervals of 200 trials. As can be seen from the figure, the adaptive selection of tours converges to a near-optimal one in the process of learning. At the beginning, the tour length is comparable to that obtained by the exhaustive search for two-bond rearrangements. As learning advances, competition and ''mating'' of the tour segments in the tours used for comparison are effectively taking place through modification of synaptic strengths — this leads to tours with shorter and shorter tour lengths. The improvement due to learning is quite drastic: the slow learning process reduces the tour lengths obtained at the begining of the learning process by as much as 40%. The difference between the average tour length and the shortest one in a given interval can be thought of as a measure of the effective ''temperature,'' in analogy with OSA; during the course of the simulation the effective temperature decreases as the learning process converges. The dependence of the performance of the algorithm on the learning rate α is illustrated in Fig. 2, in which successive average tour lengths in the intervals of 200 trials are plotted for $\alpha=0.03$, 0.12, 0.48, and 1.92. As one might expect, better results are achieved for slower learning rates (small α), but with longer computer time. The longest optimization with $N=400$ and $\alpha=0.03$ requires about 20 min on a 200 MHz SGI Power Challenge L.

To get a comprehensive picture of the performance of this algorithm and its dependence on the parameters, we have performed extensive simulations for $N=25$, 50, 100, 200, and 400 with the number of samples equal to 800, 400, 200, 100, and 50, respectively. Large numbers of samples are needed for small-size systems to obtain more reliable average tour lengths, because of the larger statistical fluctuations

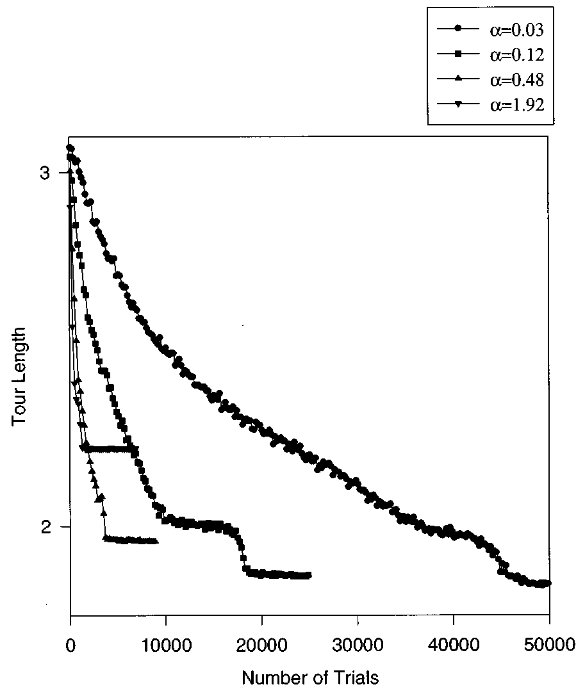

FIG. 2. Average tour length vs number of trials for the optimizations with $\alpha=0.03$, 0.12, 0.48, and 1.92. The simulations use the same 200-city sample for which the results shown in Fig. 1 are obtained.

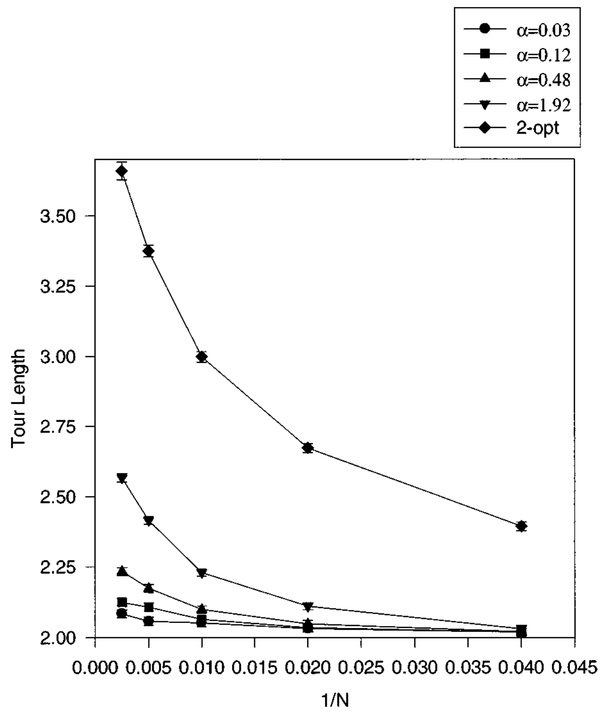

FIG. 3. Tour length vs $1/N$ (N is the number of cities) for the random distance TSP. The data on the top of the figure are obtained using the iterative improvement method of exhaustive search for two-bond rearrangements; below it, in order of decreasing tour lengths, are a series of the best tour lengths obtained using $\alpha=1.92$, 0.48, 0.12, and 0.03. Each data point represents an average over 800, 400, 200, 100, and 50 samples for $N=25$, 50, 100, 200, and 400 cities, respectively.

in the best tour lengths for smaller sample sizes. In each simulation both the shortest tour length and the number of trials needed to reach the shortest tour length are recorded. Note that the actual CPU used for a given number of trials n is not proportional to n, because many two-bond rearrangements are needed at the start of the learning process and almost no two-bond rearrangements are needed at the end of the process. The average best tour lengths for these simulations are listed (together with the average number of trials needed to reach the best lengths) in Table I and shown in Fig. 3.

We now compare our results with the known results. Krauth and Mézard [11], used the Lagrangian one-tree relaxation of Held and Karp [12] to obtain a lower bound for the best tour length and the Lin-Kernighan algorithm [13] to obtain an upper bound. The performance of the Lin-Kernighan algorithm depends somewhat on implementation. Better results are obtained in the implementation presented in Ref. [7]. The Lin-Kernighan algorithm is generally regarded as one of the best algorithms for the TSP [7] (in particular for the random distance TSP, where many other algorithms, including simulated annealing, do not give good results). Our results are generally better than those obtained

TABLE I. Average best tour length and average number of trials needed to reach the best tour length for a set of learning rate α and for the number of cities $N=25$, 50, 100, 200, and 400. For comparison, the results obtained using the exhaustive search for two-bond rearrangements (2-opt) are also listed in the table.

	Best tour length obtained and number of trials required				
N/α	0.03	0.12	0.48	1.92	2-opt
25	2.019±0.013	2.019±0.013	2.020±0.013	2.030±0.013	2.394±0.015
	1207	463	169	65	
50	2.032±0.013	2.035±0.013	2.048±0.013	2.111±0.013	2.673±0.016
	7008	2004	571	174	
100	2.052±0.013	2.064±0.013	2.099±0.013	2.230±0.013	2.998±0.019
	19044	5205	1466	418	
200	2.058±0.014	2.108±0.014	2.174±0.014	2.415±0.014	3.375±0.021
	42683	11731	2948	894	
400	2.084±0.014	2.125±0.013	2.234±0.014	2.567±0.015	3.659±0.030
	85775	21999	7107	1891	

by the Lin-Kernighan algorithm implemented by Krauth and Mézard (see Fig. 1. of Ref. [11]) even with $\alpha=0.48$. It is comparable with the Lin-Kernighan results of Ref. [7], with $\alpha=0.03$. Note that our results for $N=25$ and 50 using $\alpha=0.03$ and 0.12 are almost the same as the corresponding lower bounds, while our results for $N=100$, 200, and 400 using $\alpha=0.03$ are within 5% of the corresponding lower bounds (see Fig. 1 of Ref. [11]). In contrast, the results of simulated annealing using two-bond rearrangement are 12% and 37% above the lower bounds for $N=100$ and $N=316$, respectively [7]. With smaller learning rate α we can get even better results. This shows that a simple learning strategy can lead to a very good optimization algorithm. However, like the other algorithms which involve stochastic searches, our algorithm is slow compared with the deterministic search algorithm like the Lin-Kernighan algorithm. To illustrate the CPU usage of our algorithm, we monitor the CPU used in the optimization of a 200-city sample. The best tour lengths obtained for a given amount of CPU used are shown in Fig. 4. For comparison, we also show the results from the Lin-Kernighan algorithm. Here we use the shortest tour length of n independent Lin-Kernighan runs (starting from different initial configurations) with $n=1$, 2, 4,. . . , and 1024. The CPU used is n times the CPU used for the single Lin-Kernighan run, which averages about 0.31 s on the SGI Power Challenge. For shorter CPU time used, the Lin-Kernighan algorithm is clearly superior. However, given a longer CPU time, our learning algorithm can give better results.

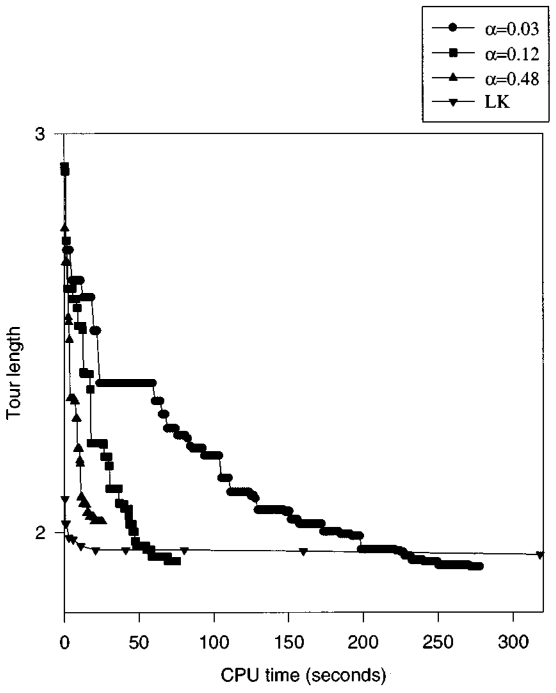

FIG. 4. Best tour length (L) obtained vs CPU time (of SGI Power Challege L) t used. $L(t)$ represents the best tour length obtained before time t. The optimizations are done with $\alpha=0.03$, 0.12, and 0.48, and on the same 200-city sample for which the results shown in Fig. 1 is obtained. For comparison, the best tour length obtained using multiple Lin-Kernighan runs is also shown. The number of Lin-Kernighan runs used is 1, 2, 4, 8,. . . , and 1024, respectively.

In conclusion, we have demonstrated how a complex optimization problem can be solved by a simple learning strategy of trial and adaptation. The learning process is quite similar to the evolution process in genetic algorithms. As in GA, the algorithm has the advantage that it is based on global searches in configuration space, where configurations far apart in configuration space are searched. But instead of keeping many tour configurations explicitly as in GA's we use ''synaptic strengths'' to generate tour configurations probabilistically. Thus many tour configurations are implicitly kept for effective mutation and mating through the updating of the ''synaptic strengths.'' The version of the learning algorithm presented in this paper for solving TSP may not be competitive, but a better algorithm can be derived using more sophisticated local search methods (such as three-bond and four-bond moves for TSP). What we have shown is that the learning process based on gradual changes on ''synaptic strengths'' can greatly improve the results obtained by the corresponding local search methods. We believe that this learning strategy will be very valuable for other optimization problems, in particular the ones where sophisticated local search algorithms have not been found. Currently we are applying the learning algorithm to find the ground state of spin glasses. Preliminary results show that this simple learning strategy is also very effective in the spin glass problem.

We thank C. Jayaprakash for a critical reading of the manuscript and for many helpful comments and suggestions.

[1] D. G. Bounds, Nature **329**, 215 (1987).
[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Science **220**, 671 (1983).
[3] J. Lee and M. Y. Choi, Phys. Rev. E **50**, R651 (1994).
[4] J. H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975).
[5] R. M. Brady, Nature **317**, 804 (1985).
[6] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, Parallel Comput. **7**, 65 (1988).
[7] D. S. Johnson and L. A. McGeoch, in *Local Search in Combinatorial Optimization*, edited by E. H. L. Aarts and J. K. Lenstra by (John Wiley and Sons, New York, in press).
[8] J. J. Hopfield and D. W. Tank, Science **233**, 625 (1986).
[9] K. Chen (unpublished).
[10] S. Lin, Bell Syst. Tech. J. **44**, 2245 (1965).
[11] W. Krauth and M. Mézard, Europhys. Lett. **8**, 213 (1989).
[12] M. Held and R. M. Karp, Oper. Res. **18**, 1138 (1970).
[13] S. Lin and B. W. Kernighan, Oper. Res. **21**, 498 (1973).