# Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics

*Olivier Martin, Steve W. Otto, Edward W. Felten*

Oregon Graduate Institute
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 91-015

June, 1991

# Large-Step Markov Chains for the TSP Incorporating Local Search Heuristics*

Olivier Martin
Department of Physics,
City College of City University of New York,
New York, NY, 10031, USA

Steve W. Otto
Dept of Computer Science and Engineering,
Oregon Graduate Institute of Science and Technology,
19600 NW von Neumann Dr, Beaverton, OR, 97006, USA
otto@cse.ogi.edu

Edward W. Felten
Dept of Computer Science,
University of Washington, Seattle, WA, 98195, USA

Keywords: TSP, simulated annealing, Markov, Optimization

July 3, 1990 revised April 12, 1991

**Abstract**

We consider a new class of optimization heuristics which combine local searches with stochastic sampling methods, allowing one to iterate local optimization heuristics. We have tested this on the Euclidean Traveling Salesman Problem, improving 3-opt by over 1.6% and Lin-Kernighan by 1.3%.

## 1 Introduction

Given $N$ cities labeled by $i = 1, N$, separated by distances $d_{ij}$, the Traveling Salesman Problem (TSP) consists in finding the shortest tour, i.e., the shortest

*This work was supported in part by the grants DOE-FG03-85ER25009 and NSF-ECS-8909127, and by a grant from the PSC-CUNY Research Award Program. Correspondence regarding this work should be addressed to S. Otto closed path visiting every city exactly once. To be specific, we will consider the symmetric TSP where $d_{ij} = d_{ji}$, but our method generalizes to the asymmetric case also. The problem of finding the optimal tour is a difficult one as the TSP is NP complete. There are a number of exact methods that are guaranteed to find the exact optimum in a bounded number of steps. These include branch and bound methods, and plane-cutting methods. See Lawler et. al. for an overview [9]. These methods have progressed tremendously in the last ten years, so that instances with $N$ of several thousand have been solved to optimality [14]. In such algorithms, as well as in many practical situations, it is useful to also have nearly optimal solutions. These are typically obtained by heuristic approaches which do not guarantee that the optimal solution will be found. There are many of these heuristic methods for the TSP: tour construction, local search [10, 11], simulated annealing [8, 1], genetic algorithms [13], etc. In this note, we present a methodology for improving local search methods, and this leads to a new heuristic for the TSP that, in particular, surpasses Lin-Kernighan (L-K) [11].

## 2 Local Opt Searches

In a local opt search method, one first defines a notion of neighborhood (i.e., a topology) on the set of all tours. For instance, one might define the neighborhood of a tour, $T$, to be all those tours which can be obtained by changing at most $k$ edges of $T$. A tour is said to be locally optimal if no tour in its neighborhood is shorter than it. One can search for local $k$-opt tours by starting with a random tour $T_1$ and constructing a sequence of tours $T_1$, $T_2$, ... Each tour is obtained from the previous one by performing a $k$-change, i.e., by deleting $k$ links and reconnecting the loose ends so as to still have a tour. The $k$-change is required to decrease the length of the tour. Eventually the process stops because one has reached a tour for which there is no possible improvement under a $k$-change. Lin [10] studied the case of $k$=2 and $k$=3, and showed that one could get quite good tours quickly. In order to find the globally optimal tour, he suggested repeating the search from random starts many times until one was confident all the locally optimal tours had been found. Perhaps the most widely referred to optimization heuristic is the Lin-Kernighan algorithm. It is a variable-$k$ neighborhood search, and it is the benchmark against which most heuristics are tested. It is significantly better than 3-opt: on average L-K-opt tours are shorter than 3-opt tours, and for a given instance of the TSP, there are many fewer L-K-opt tours than there are 3-opt tours. Generally, one can parameterize the number of local-opt tours by $exp(\alpha N)$, and then $\alpha$ for 3-opt is larger than for L-K-opt. Because of this exponential growth with $N$, finding the globally optimal tour by repeated random starts becomes unmanageable at large $N$ for any local search method. If one is limited to a few hours on a workstation, searching for all 3-opt (resp. L-K) local-opt tours becomes unmanageable at $N \approx 80$ (resp. $N \approx 200$).

Given that one really does want to tackle larger problems, there are two natural approaches. First, one can try to extend the neighborhood which L-K considers, just as L-K extended the neighborhood given by 3-changes. Second, instead of sampling the local opt tours in a random way (as is done by repeatedly using random starts), it might be better to obtain locally optimal tours using a biased sampling which would favor the shorter tours. Our algorithm does both these things and indeed beats L-K. Furthermore, the general methodology can be used to combine arbitrary local search methods with stochastic sampling methods, leading to an improvement over both underlying algorithms.

## 3 Markov Chains and Simulated Annealing

Given that any local search method will stop in one of the many locally optimal solutions, it may be useful to find a way for the search to continue by temporarily allowing the tour length to increase. This leads to the popular method of simulated annealing [8, 1]. Simulated annealing is a general purpose algorithm suitable for many optimization problems. It is based on using what is called a Markov chain to sample solutions iteratively and stochastically. For the case of the TSP, one starts by constructing a sequence of tours $T_1$, $T_2$, etc... Each step of this chain is obtained by doing a $k$-change (moving to a neighboring tour). Usually, $k$ is 2 or 3. If the tour length decreases, the change is accepted; if the tour length increases, the change is rejected with some probability (in which case one simply "repeats" the old tour at that step). This stochastic construction of a sequence of $T$'s can be viewed as an extension of the above local search to include "noisiness." Because increases in the tour length are possible, this chain never stops. For many such Markov chains, it is possible to show that given enough time, the chain will visit all tours, and that for very long chains, the $T$'s appear with a calculable probability distribution. Markov chains are closely inspired by physical models where the chain construction procedure is called a Monte Carlo. The stochastic accept/reject part simulates a random fluctuation due to temperature effects, and the temperature is a parameter which measures the amount of bias. To avoid wasting time sampling bad solutions, one decreases the temperature with time according to an annealing schedule, and this is why the algorithm is called simulated annealing.

If the temperature is taken to zero too fast, the effect is essentially the same as setting the temperature to zero immediately, and then the chain gets trapped forever at a locally optimal solution just as in local searches. There are theoretical results on how slowly the annealing has to be done to be sure that one reaches the global optimum, but in practice the required running times are astronomical. Nevertheless, simulated annealing is a standard method which is competitive for a number of optimization problems [4, 5, 6]. For the TSP, it is significantly slower than Lin-Kernighan, but it has the advantage that one can run for long times and slowly improve the quality of the solutions, eventually getting better results than L-K. (See for instance the studies of Johnson et. al. [4].) The improvement is due to the better sampling of the short length tours: tours which are not near the minimum length are ignored. Simulated annealing tries to improve an already very good tour, one which probably has many links in common with the exact optimum. The standard Lin-Kernighan algorithm, by contrast, continually restarts from scratch, throwing away possibly useful information.

## 4 Iterated Local Opt

In this section, we show how to combine the advantages of local search heuristics with stochastic sampling methods. Although the methodology is general, we explain our approach in the context of the TSP only.

A drawback of simulated annealing is that it does not take advantage of local opt heuristics. This means that instead of sampling local opt tours as does multiple runs of L-K, the Markov chain of simulated annealing samples all tours. It would be a great advantage to be able to restrict the sampling to the local opt tours only. Then the bias provided by the Markov chain would enable one to sample the shortest local opt tours more efficiently than by using repeated random starts of local opt. To do this, one has to find a way to go from one local opt tour, $T_n$, to another local opt tour, $T_{n+1}$, and this is the heart of our procedure. We propose to do a change on $T_n$, which we call a "kick." This can be a random $p$-change, but we will choose something smarter than that as explained at the end of this section. Follow this kick by the local opt tour improvement heuristic until a new local opt tour, $A_n$, is reached. Then accept or reject $A_n$ depending on the increase or decrease in tour length compared to $T_n$. The result is $T_{n+1}$. Since there are often many changes in going from $T_n$ to $T_{(n+1)}$, we call this method a "large-step markov chain." It can also be called "iterated local opt," but it should be realized that the difficulty is precisely in finding a way to iterate! The algorithm is far better than the small step Markov chain methods like simulated annealing because the accept/reject procedure is not implemented on the intermediate tours which are almost always of longer length. Instead, the accept/reject happens only after the system has returned to a local minimum. The method directly steps from one local minimum to another.

At this point, let us mention that this method is no longer a true simulated annealing algorithm. That is, the algorithm does NOT correspond to the simulation of any physical system undergoing annealing. The reason is that a certain symmetry property, termed detailed balance, is not satisfied by the large-step algorithm. One consequence of this is that the "temperature" parameter used in the accept/reject procedure no longer plays the role of a true physical temperature – instead it is merely a parameter which controls the bias towards the optimum.

We have found that in practice, the above methodology provides an efficient sampling of the local opt tours. There are a number of criteria which need to be met for the biased sampling of the Markov chain to be more efficient than plain random sampling. These conditions are satisfied for the TSP, and more generally will be so whenever local search heuristics are useful. Let us stress that this large step Markov chain approach is extremely general, being applicable to any optimization problem where one has local search heuristics. It enables one to get a performance which is at least as good as local search, with substantial improvements over that if the sampling can be biased effectively. Finally, although the method is general, it can be adapted to match the problem of interest through the choice of the kick. We will now discuss how we have choosen the kick for the TSP.

Consider the case where the local search method is 3-opt. If we used a kick consisting of a 3-change, 3-opt would very often simply bring us back to the starting tour with no gain. Thus it is probably a good idea to use at least a 4-change for the kick when the local search is 3-opt. For more general local search algorithms, a likely choice for the kick is a $k$-change which does not occur in the local search. Interestingly, it turns out that there is one kick choice which is both natural and effective for 2-opt, 3-opt, and L-K. To see this, it is useful to go back to the work of Lin and Kernighan. In their paper, they define "sequential" changes, and they also show that if a tour can be improved, one can force all the partial gains during the $k$-change to be positive. A consequence of this is that the check-out time for sequential $k$-changes can be completed in $O(N)$ steps. It is easy to see that all 2 and 3 changes are sequential, and that the first *non*-sequential change occurs at $k$=4 (figure 2 of the Lin and Kernighan paper). We call this graph a "double-bridge" change because of what it does to the tour. First it does a 2-change which disconnects the tour; then it does a second 2-change which reconnects the two parts by creating a bridge. Both of the 2-changes are bridges in their own way, and the double-bridge change is the only non-sequential 4-change which cannot be obtained by composing changes which are both sequential and leave the tour connected. Thus the double bridge is the most natural kick choice for any local search method which considers only sequential changes. L-K is of this type: all allowed changes are sequential. It considers many changes with $k$ greater than 3, but it misses double-bridges. Thus one expects that most of what remains in excess length using L-K might be removed with our extension. The results below indicate that this is indeed the case. Note that if we included this double-bridge change in the definition of the neighborhood for a local search (that is, extending the L-K heuristic), check-out time would require $O(N^2)$ steps (essentially a factor $N$ for each bridge). Rather than doing this change as part of the local search, we include such changes stochastically as our kick.

We have implemented iterated local opt using "double-bridges" for the kicks, and 3-opt and L-K for the local searches. The pseudo-code for the algorithm appears in Table 1.

```
input instance ($N$ and $d_{ij}$)              input the instance
input temperature                              for accept/reject
input $M$                                      length of Markov chain
input $T_{init}$                               starting tour
$T_1$ = localOpt( $T_{init}$ )                 $T_1$ is first element of chain
do i=1 to $M$                                  generate chain
        $B_i$ = doubleBridge( $T_i$ )           apply kick
        $A_i$ = localOpt( $B_i$ )               opt the result
        $T_{i+1} = A_i$ or $T_{i+1} = T_i$      accept/reject step
```

Table 1: A pseudo-code specification of the large-step markov chain algorithm.

There is some freedom in how one chooses the double-bridges. The simplest procedure is to do that selection at random. First, choose two bonds at random, and do the corresponding 2-change which disconnects the tour. Second, choose again at random two other bonds. If the corresponding 2-change reconnects the tour, one has a double-bridge and one is done. If not, start over, choosing another first bridge. We use this procedure with one modification: we accept only bridges shorter than some given length. This constraint guarantees that the kick will not increase the tour length too much, and thus it tends to increase the acceptance rate of the large step Markov chain. Nevertheless, we have found in practice that the performance of the iterated local opt is not sensitive to this maximum length as long as it is at least a few times the average nearest-neighbor length. Note that almost all the CPU time is spent doing the local opt, so the selection of a kick can be quite involved without slowing down the algorithm.

## 5 Results

We first implemented the large step Markov chain with the local search being 3-opt. We generated TSP instances by randomly generating $N$ points in a unit square. For $N$ up to 200, we were able to determine the optimal tour using a branch and bound program. We then ran the large step Markov chain algorithm. In all cases, it found the optimal tour rather quickly. The average time for finding the optimum had some variation from instance to instance, so we have averaged our times over our instances. For N=100, the optimum was found on average in a bit over two minutes on a SUN-3, while for N=200, the average time was 1.7 hours. For larger instances, we used problems which had been solved to optimality by other groups. We ran our program on the Lin-318 instance solved to optimality by Crowder and Padberg [3]. Our iterated 3-opt found the optimal tour on each of five separate runs, with an average time of 19 hours on the SUN-3. We also ran on the AT&T-532 instance solved to optimality by Padberg and Rinaldi [14]. By using a post-reduction method inspired by tricks explained in the Lin-Kernighan paper, the program found the optimum solution in 94 hours (Martin [12]). Repeated runs of ordinary 3-opt from random starts did not solve the Lin-318 nor the AT&T-532 problems to optimality even with significantly more CPU time.

It is of interest to ask what is the expected excess tour length for very large problems using our method. We have run on large instances of cities randomly distributed in the unit square. Ordinary 3-opt gives an average length 3.6% above the Held-Karp lower bound, whereas the iterated 3-opt leads to an average of less than 2.0% above that bound. Thus we see that without much more algorithmic complexity, our use of double-bridges improves 3-opt very significantly.

In our preprint [12], we suggested that such a dramatic improvement should also carry over to the case where L-K is used instead of 3-opt. Johnson [7], Cook et. al. [2], and ourselves have now investigated the improvement of iterated L-K over repeated L-K. Their implementation of the large step algorithm differs slightly from ours: they do not impose any constraints on the double-bridges, choosing them randomly, and they take the temperature to be zero. In all implementations, iterated L-K is much better than ordinary L-K; it is able to find the solution to Lin-318 in minutes, and the solution to AT&T-532 in an hour on a SUN-Sparcstation. (Again, L-K repeated from random starts was unable to solve these problems to optimality.) At the TSP-90 conference held at Rice University in April 1990, we ran on a 783 city instance provided by the conference organizers. Our code found the optimum in 6 hours on a Sparcstation, while the code of Johnson and that of Cook, Applegate and Chvatal found this optimum even faster. Finally, for large instances (randomly distributed cities in a square), Johnson [7] finds that iterated L-K leads to an average excess length of 0.84% above the Held-Karp bound. Previously it was expected that the exact optimum was 1% or more above the Held-Karp bound, but iterated L-K disproves this conjecture.

One of the most interesting results of these experiments is that for "moderate" sized problems (such as the AT&T 532 or the 783 instance mentioned above), no "annealing" seems to be necessary. Just setting the temperature to zero (no uphill moves at all) gives an algorithm which can often find the exact optimum. The implication is that, for the large step Markov chain algorithm, the effective tour length landscape has one or only a few local minima! Almost all of the L-K local minima have been modified to saddle points by extending the neighborhoods of the tours.

## Acknowledgements

We thank David Johnson and Bill Cook for sharing their results with us and for discussions.

## References

[1] V. Cerny. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J. of Opt. Theo. and Appl.*, 45:41, 1985.

[2] W. Cook, V. Chvatal, and D. Applegate. Private communication.

[3] H. Crowder and M.W. Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26:495–509. A discussion of this also appears in Chapter 9 of the book by Lawler, et. al.

[4] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part iii (the traveling salesman problem). In preparation.

[5] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part ii (graph coloring and number partitioning). To appear in Operations Research.

[6] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part i (graph partitioning). To appear in Operations Research.

[7] D.S. Johnson. Local optimization and the traveling salesman problem. In *17'th Colloquium on Automata, Languages, and Programming.* Springer-Verlag, 1990. Also, private communication.

[8] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671, 1983.

[9] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem.* John Wiley & and Sons, 1984.

[10] S. Lin. Computer solutions of the traveling salesman problem. *Bell Syst. Tech. J.*, 44:2245, 1965.

[11] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Oper. Res.*, 21:498, 1973.

[12] O. Martin, S. Otto, and E. Felten. Large-step markov chains for the traveling salesman problem. Preprint.

[13] H. Muhlenbein, M. Georges-Schleuter, and O. Kramer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65, 1988.

[14] M.W. Padberg and G. Rinaldi. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.*, 6(1):1–7, 1987.