PERGAMON

Neural Networks 13 (2000) 283–285

Neural
Networks

www.elsevier.com/locate/neunet

Neural Networks Letter

Improving local minima of Hopfield networks with augmented Lagrange multipliers for large scale TSPs

M. Martín-Valdivia[a,*], A. Ruiz-Sepúlveda[b], F. Triguero-Ruiz[b]

[a]*Departamento de Informática, University of Jaén, Campus Las Lagunillas, s/n E-23071, Jaen, Spain*
[b]*Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E-29071, Malaga, Spain*

PERGAMON

Neural Networks 13 (2000) 283–285

Neural Networks

www.elsevier.com/locate/neunet

Neural Networks Letter

Improving local minima of Hopfield networks with augmented Lagrange multipliers for large scale TSPs

M. Martín-Valdivia[a,*], A. Ruiz-Sepúlveda[b], F. Triguero-Ruiz[b]

[a]*Departamento de Informática, University of Jaén, Campus Las Lagunillas, s/n E-23071, Jaen, Spain*
[b]*Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E-29071, Malaga, Spain*

Received 23 February 1999; accepted 22 November 1999

Keywords: Optimization; Hopfield model; Traveling salesman problem; Lagrange multipliers

1. Introduction

The Hopfield model has been used to solve optimization problems and, mainly, to solve the Traveling Salesman Problem (TSP). Since 1985 (Hopfield & Tank, 1985), when it was first used, much research has been aimed at applying the Hopfield network to solve this kind of problem. The Hopfield model converts an optimization problem with constraints into an equivalent unconstrained optimization problem using the penalty functions method.

Recently, Li (1996) proposed a new approach to solve the TSP with a Hopfield network based on augmented Lagrange multipliers (ALH, Augmented Lagrange Hopfield). This method, which combines Lagrangian multipliers and penalty functions to solve an optimization problem with constraints, overcomes the problems that both methods present when used separately. By using Lagrangian multipliers, constraints are satisfied without the need to send the penalty terms to infinity, and with the penalty terms the zigzagging problem of the Lagrange multiplier method is alleviated. However, though it is not possible to improve the convergence rate, because the ALH method obtains 100% of valid tours, the solutions found continue to correspond to local minima. Moreover, with the increase in the network scale, the local minima problem becomes much worse.

In order to avoid the local minima problem, many researchers have come to use different heuristic search techniques. Some such techniques use local minima escape (LME) algorithms (Peng, Gupta & Armitage, 1996), so that whenever the network is trapped into a local minimum state, the LME algorithm can be used to find a new state which is at the same or lower energy level than the present local minimum state.

In this work, we propose a local minima escape algorithm which, instead of applying it to the original Hopfield model (with penalty functions), takes as its base the ALH method (Li, 1996). To test the efficiency of the algorithm, we apply it to the TSP with a large number of cities (51 and 101 cities).

2. The LME algorithm for the ALH model

In relation to convergence and solution quality, the ALH method surpasses the Hopfield model or its modified versions based on the penalty functions method, but, just as occurs with other methods, when the network size is increased, the solutions represent local minima whose quality is worse every time.

In order to improve the local minima quality for large-scale problems we propose a local minima escape algorithm for a Hopfield network based on the augmented Lagrange method (LME-ALH), so that whenever the network is trapped into a local minimum state, the LME algorithm can be used to find a new state which is at the same or lower energy level than the present local minimum state. The LME-ALH algorithm uses the ALH equations (Li, 1996) to complete an iteration. The algorithm operates in the following way: Let H be a Hopfield network. The network is initialized with a valid tour that corresponds to a stable state for the network (a local minimum). Random noise is then added into the neurons in order to disturb the network and force it to evolve to another stable state H' (a new valid tour). If the new state H' has a lower energy level than H, H' is taken as the initial tour and the process is repeated. Otherwise, the process is repeated but returns to

* Corresponding author. Tel.: +34-953-212202; fax: +34-953-212222.
E-mail address: maite@ujaen.es (M. Martín-Valdivia).

PII: S0893-6080(00)00012-5

using H as the initial state. The LME-ALH algorithm consists in repeating this process a fixed M number of times, so that, for each iteration, the best tour found up to that moment is taken as the initial tour. After repeating the process M times, the final solution for a given run will be the tour with the shortest length found during all the iterations.

In the first iteration (for $M = 1$) the network starts from the default tour as the valid initial tour. The default tour is simply the set of cities ordered according to the value of the city label. Thus, for example, for a network with 10 cities, the default tour is 1–2–3–4–5–6–7–8–9–10. As a rule, for a TSP of N cities, the state of the $N \times N$ neurons is initialized in the following way:

$$v_{Xi} = \begin{cases} 1 & \text{if } X = i \\ 0 & \text{if } X \neq i \end{cases} \quad (1)$$

This initialization represents a stable state, so noise, δ, must be introduced into the neurons in order to disturb the network and force it to evolve to another stable state with a lower energy level.

$$v_{Xi} = v_{Xi} + \delta \quad (2)$$

The algorithm operates in the following way: the best solution found up to that moment is always stored in the matrix $S_{N\times N}$. $f(X)$ represents the function that calculates the total distance for a given tour.

1. Initialize the solution tour as the default

$$s_{Xi} = \begin{cases} 1 & \text{if } X = i \\ 0 & \text{if } X \neq i \end{cases}$$

$$D_s = f(S)$$

2. Repeat M times
   2.1. $v_{Xi} = s_{Xi} + \delta$
   2.2. $u_{Xi} = -\tau Ln((1 - v_{Xi})/v_{Xi})$
   2.3. Make the network proceed until a new local minimum is found, that is, apply the ALH algorithm
   2.4. $D_v = f(V)$
   2.5. If $D_v < D_s$ then
   $D_s = D_v$
   $s_{Xi} = v_{Xi}$
3. Show the final solution: the tour found is in the matrix S whose total length is D_s

δ is a uniformly distributed random number that is generated at the start of each iteration. This random number is used to introduce noise into the neurons in order to disturb the network when a stable state is reached that corresponds to a local minimum. D_v contains the tour length found for each iteration, whereas D_s contains the length of the best solution found up to that moment. For each iteration, v_{Xi} store the state of the neurons while the network progresses, whereas s_{Xi} always store the state of the neurons for the best tour found up to that moment.

3. Results

To test its performance, the LME-ALH algorithm are applied to the TSP with 51 and 101 cities. Each test case is run 10 times with different initializations. The city coordinates and the optimal tours for these two problems come from the TSPLIB library collected by Bixby and Reinelt. More specifically, the files "eil51.tsp" and "eil101.tsp" contain the data set for the TSP with 51 and 101 cities respectively, whereas the files "eil51.opt.tour" and "eil101.opt.tour" contain the optimum tour with its total length (426 for 51 cities and 629 for 101 cities).

In every iteration, the v variables are initialized with the best tour found up to that moment (in the first iteration of each run, $M = 1$, v is initialized with the default tour). Then, a uniformly distributed random noise, δ, is added to every v. The number of iterations per run is fixed at 5 ($M = 5$.) In general, M can take any value but it should be noted that the greater it is, the better the solution quality and the longer the convergence time. The rest of the variables for the LME-ALH algorithm are initialized as follows: $\beta_1 = \beta_2 = 100$, $\beta_3 = \beta_4 = 1$, $\beta_5 = 0$, $\mu = 100$, $T = 10^7$. For the TSP with 51 cities, $D = 10^5$, $\lambda = 10^4$, and for the TSP with 101 cities, $D = 10^4$, $\lambda = 10^3$.

Finally, the noise δ that is added to the v variables when the network is trapped into a local minimum (it has reached a stable state) affects solution quality quite strongly. To determine which is the best interval it must vary, the influence of this parameter has been investigated using different intervals in which δ is randomly distributed. For each problem (TSP with 51 and 101 cities) a uniformly distributed random noise is added with $\delta \in [0, K]$ and are carried out 10 runs with this interval. We have tested 10 different intervals starting with $K = 0.1$ and ending in $K = 1.0$, so that K varies according to $K \leftarrow K + 0.1$:

For $K = 0.1$ to $K = 1.0$
  Apply the LME-ALH algorithm during 10 runs with $\delta \in [0, K]$
  $K \leftarrow K + 0.1$
End_for

Table 1
Data obtained for the TSP with the LME-ALH algorithm

Number of cities	Average distance	Error (%)	Minimum distance	Maximum distance
51	478.9	12.4	467	493
101	828.8	31.8	778	875

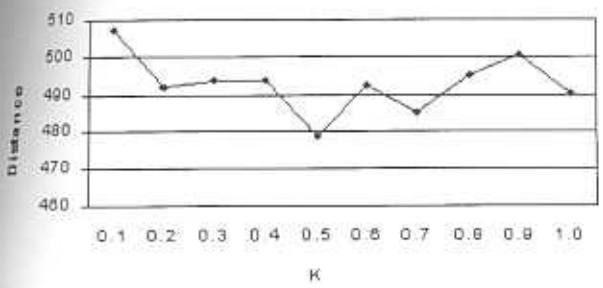

Fig. 1. Average distance for the TSP with 51 cities varying the intervals in which δ is distributed.

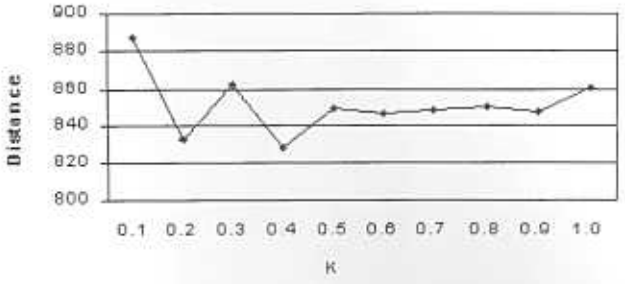

Fig. 2. Average distance for the TSP with 101 cities varying the intervals in which δ is distributed.

Figs. 1 and 2 summarize the results obtained for the TSP with 51 and 101 cities, respectively, and they show the average tour length found for different values of K. According to Fig. 1, the best results for the TSP with 51 cities are obtained when δ is a real number uniformly distributed in the interval [0,0.5], whereas Fig. 2 shows that the behavior of the network is better in the interval [0,0.4] for the TSP with 101 cities. The results in Table 1 refer to these intervals.

For the TSP with 51 cities, the shortest tour is 9.6% longer than the optimum tour whereas the worst solution is 15.7% longer than the optimum. With 101 cities, the error rates, with regard to the optimum tour, are 23.7 and 39.1% for the minimum and maximum distance reached, respectively.

4. Conclusions

In this work, we have presented an LME-ALH algorithm to escape the local minima for a Hopfield network that uses the augmented Lagrange multiplier method (ALH). The ALH method combines the penalty functions method (which is the basis for most of the methods published up until now) and Lagrange multipliers simultaneously, in the energy function of the Hopfield network to solve the TSP. In this way, the problems that both methods present when they are used separately are solved. The ALH method is quite efficient when dealing with small-scale networks [3]. However, for large-scale problems, the solutions reached with the ALH algorithm correspond to local minima. In order to improve solution quality, the LME-ALH algorithm combines the ALH method with a local minima escape algorithm that introduces noise into the neuron state to disturb the network.

References

Bixby, B., & Reinelt, G. Traveling Salesman Problem Library (on line). Available ftp://elib.zib-berlin.de/pub/mp-testdata/tsp/tsplib/tsp.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decisions optimization problems. *Biological Cybernetics, 52*, 141–152.

Li, S. Z. (1996). Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers. *IEEE Transactions on Neural Networks, 7*, 1507–1516.

Peng, M., Gupta, N. K., & Armitage, A. F. (1996). An investigation into the improvement of local minima of the Hopfield Network. *Neural Networks, 9*, 1241–1253.