Bull. Nov. Comp. Center, Comp. Science, 30 (2010), 89–94

A parallel algorithm for solving the traveling salesman problem by a recurrent neural network

M.S. Tarkov, G.A. Dugarov

Abstract. A parallel algorithm for solving the traveling salesman problem by the recurrent Wang neural network in conjunction with the WTA ("Winner Takes All") principle is proposed. This algorithm is preferable when it is necessary to solve the problem with sufficient accuracy in a lesser time.

1. Introduction

Conventional sequential computing technologies do not allow us to find a solution of discrete-large scale optimization problems. The branch-and-bound algorithm (the Little method) [1, 2] is one of the basic techniques to solve them. Its tree-like structure suggests a parallel implementation because computations in different branches can be independently realized. Nevertheless, the algorithm parallelization can be very complicated because of a huge capacity of utilized memory and the necessity to balance a computing load among processors [3]. On the other hand, the massive parallelism of data processing on a neural network [4–12] allows one to consider it as a high efficient and reliable tool for solving complicated optimization problems. Recurrent neural networks are most interesting for this application.

2. A neural network statement of the traveling salesman problem

The traveling salesman problem (TSP) can be formulated as an optimization problem

$$\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij} \to \min, \tag{1}$$

subject to

$$\begin{aligned} &x_{ij} \in \{0,1\}, \quad x_{ii} = 0, \\ &\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,\ldots,n, \\ &\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,\ldots,n, \end{aligned} \tag{2}$$

where C_{ij}, $i \neq j$, is the cost of the assigning entry i to the position j, x_{ij} is a decision variable: if the entry i is assigned to the position j then $x_{ij} = 1$, otherwise $x_{ij} = 0$.

To solve this problem, J. Wang [10] proposed a recurrent neural network which is described by the following differential equation:

$$\frac{\partial u_{ij}(t)}{\partial t} = -\eta\Big(\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t) - 2\Big) - \lambda C_{ij} \exp\Big(-\frac{t}{\tau}\Big), \qquad (3)$$

where $x_{ij} = f(u_{ij}(t))$, $f(u) = (1 + \exp(-\beta u))^{-1}$. In this network, a matrix of neurons of size $n \times n$ is used, and the neurons are communicated in rows and columns.

A difference version of equation (3) is

$$u_{ij}^{t+1} = u_{ij}^{t} - \Delta t\left[\eta\Big(\sum_{k=1}^{n} x_{ik}^{t} + \sum_{l=1}^{n} x_{lj}^{t} - 2\Big) - \lambda C_{ij} \exp\Big(-\frac{t}{\tau}\Big)\right], \qquad (4)$$

where Δt is a time step. The parameters Δt, η, λ, τ, and β are empirically chosen.

In [11], an algorithm using the WTA ("Winner takes all") principle for acceleration of solving equation (3) is proposed. The following algorithm [12] is a simplified version of the algorithm [11]:

0. Create the decision matrix $\|x_{ij}^0\|$ of random variables $x_{ij}^0 \in [0, 1]$.

1. Continue iterations (4) until

$$\sum_{k=1}^{n} x_{ik}(t) + \sum_{l=1}^{n} x_{lj}(t) - 2 \leq \delta, \qquad (5)$$

where δ is the accuracy of bounds (2).

2. Transform the decision matrix $\|x_{ij}\|$:

2.1. $i := i_{\text{start}}$.

2.2. Find a maximum entry $x_{i,j_{\max}}$ in the row i, $j_{\max}$ is the column number of the maximum entry.

2.3. Assign $x_{i,j_{\max}} := 1$. Make all other entries of the row i and the column $j_{\max}$ equal to zero:

$$x_{ij} := 0, \quad j \neq j_{\max}, \qquad x_{k,j_{\max}} := 0, \quad k \neq i.$$

Go to the row $i := j_{\max}$.

Repeat Steps 2.2 and 2.3 until returning to the row i_{start}.

3. Repeat Steps 1 and 2 while the return to the row i_{start} was realized earlier than n entries in the matrix $\|x_{ij}\|$ obtained value 1 (i.e., the constructed traveling salesman tour was shorter than n).

After the above algorithm execution, the 2-opt correction [13] is used to minimize the algorithm error.

3. Experiments

In [12], it is proposed to penalize unwanted assignments. To diminish the probability of an early return to the starting row i_{start}, we assume

$$C_{ij} = \begin{cases} p \cdot l_{ij}, & j = i_{\text{start}}, \\ l_{ij}, & j \neq i_{\text{start}}, \end{cases}, \quad i \neq i_{\text{start}},$$

where l_{ij} is a distance between the towns i and j and $p > 1$ is the penalty coefficient. According to [9]

$$\lambda = \frac{2(1-\alpha)\eta}{l_n^{\min} \exp(-\frac{t}{\tau})},$$

where $l_n^{\min}$ is the nth minimum value among the entries of the matrix $\|l_{ij}\|$, and the coefficient α is close to 1. To provide the algorithm efficiency, the parameter values $p \simeq 10^2 \div 10^3$, $\alpha = 0.9$, $\beta = 0.01$, $\eta = 1$, and $\tau = 1000$ were empirically chosen.

The Wang network and the branch-and-bound algorithm were tested on the TSP examples taken from the TSPLIB [14]. In Table 1, average errors of the above-mentioned techniques are shown. The errors are evaluated as coefficients $\varepsilon_W = \frac{l_W - l_{\text{opt}}}{l_{opt}}$ for the Wang network and $\varepsilon_L = \frac{l_L - l_{\text{opt}}}{l_{opt}}$ for the Little method. Here l_W and l_L correspond to tour lengths and l_{opt} is a minimum tour length.

Table 1 shows that the Little method gives more accurate solutions than the Wang network, but when the problem size n is increased, the difference is reduced. In Table 2, the Wang network and the Little method are compared by the time of solving the TSP examples. Here t_W is the time of solving with the Wang network, and t_L is the time resulted from the Little method.

Table 1. TSP solution error, %

	Problem size n		
	51	100	262
ε_W	3.84	13.18	11.44
ε_L	0.8	9.1	9.4

Table 2. TSP calculation time, s

	Problem size n		
	51	100	262
t_W	0.16	0.97	14.83
t_L	0.65	7.61	501.66

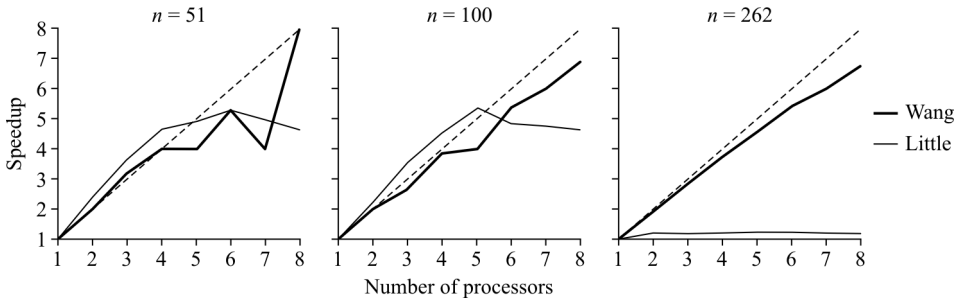
These times are realized on a single core of the processor Intel Core 2 Quad Q6600 2.4 Ghz. From Table 2, one can see that the Wang network solves the TSP quicker than the Little method.

4. Parallelizing the solution of the TSP

A parallel solution of the TSP by the recurrent neural network is based on distributing the rows of the neuron matrix among processors of a computer cluster. The rows of the matrices $\|C_{ij}\|$ and $\|x_{ij}(0)\|$ are distributed among processors. Further, the Wang network equations are solved in parallel until realizing the condition (5). After completion of the Wang network functioning, the matrix $\|x_{ij}\|$ rows are gathered in the root processor of the cluster. The WTA and the 2-opt algorithms are sequentially realized.

In parallel solving the TSP by the Little method, the master processor assigns tasks to the slave processors. The master generates a set of tasks and gathers the tour versions with the lowest estimations of length. If there is a free slave, then the master sends a task with the lowest current estimation to it. The slave splits the task obtained into two subtasks and computes the lowest estimations for them. If the estimation of a certain subtask is more or equal to the current record (a minimum estimation), then this subtask is rejected. Otherwise, a subtask is solved, and another subtask is returned to the master.

The parallel algorithms are realized on the C++ with the MPI library on a cluster with two 4-core processors Intel Core 2 Quad Q6600 2.4 GHz with RAM 4 Gb and interconnection network Gigabit Ethernet. The results of testing are presented in the figure.



Speedup of solving the TSP on the cluster

5. Conclusion

A new approach to solve the TSP problem is considered. This approach is based on application of the recurrent Wang network in conjunction with the WTA ("Winner takes all") and the 2-opt algorithms. This approach is

parallelized and realized on a multiprocessor cluster with the usage of the MPI library.

The approach is tested on the TSP examples from the TSPLIB and compared to a parallel algorithm based on a branch-and-bound approach (the Little method).

The results of the experiment show that:

1. In theory, the branch-and-bound method (the Little method) is capable to find an optimal tour, but in the method implementation it is necessary to save a great number of tour versions in a computer memory. It is not always possible because of a limited memory capacity that decreases the solution accuracy.
2. The recurrent Wang network gives less accurate solutions than the Little method, but the difference is essentially reduced with an increase in the problem size.
3. The Wang network solves the problem much quicker than the Little method.
4. For a huge TSP size, the Wang network is parallelized much better than the Little method.

Thus, for a huge TSP size, the techniques based on the Wang recurrent network are preferable if their accuracy is satisfactory. This requirement is known as principle of rational strictness [15] and rejecting the necessity of having an absolutely accurate problem solution. It is typical of the neural net applications to data processing in spacecrafts when it is necessary to solve a problem in a lesser time.

References

[1] Romanovsky I.V. Algorithms for Solving Extremal Problems. — Moscow: Nauka, 1977 (In Russian).

[2] Little J., Murty K., Sweeney D., Karel C. An algorithm for the traveling salesman problem // Operations Research. — 1963. — Vol. 11. — P. 972.

[3] Posypkin M.A., Sigal I.Kh., Galimjanova N.N. Parallel Algorithms for Discrete Optimization Problems: Computational Models, Library, Experiment Results. — Moscow: Dorodnitsin's Computing Center of RAS, 2006 (In Russian).

[4] Ossowsky S. Neural Networks for Data Processing. — Moscow: Finansy i statistika, 2002 (In Russian).

[5] Haykin S. Neural Networks. A Comprehensive Foundation. — Prentice Hall, 1999.

[6] Tarkov M.S. Neurocomputer Systems. — Moscow: INTUIT Binom. Laboratorija Znanij. — 2006 (In Russian).

[7] Melamed I.I. Neural networks and combinatorial optimization // Avtomatika i telemekhanika. — 1994. — No. 4. — P. 3–40 (In Russian).

[8] Smith K.A. Neural networks for combinatorial optimization: a review of more than a decade of research // INFORMS J. on Computing. — 1999. — Vol. 11. — No. 1. — P. 15–34.

[9] Feng G., Douligeris C., The convergence and parameter relationship for discrete-time continuous-state Hopfield networks // Proc. Intern. Joint Conference on Neural Networks. — 2001. — P. 376–381.

[10] Hung D.L., Wang J. Digital hardware realization of a recurrent neural network for solving the assignment problem // Neurocomputing. — 2003. — Vol. 51. — P. 447–461.

[11] Siqueira P.H., Steiner M.T.A., Scheer S. A new approach to solve the traveling salesman problem // Neurocomputing. — 2007. — Vol. 70. — P. 1013–1021.

[12] Tarkov M.S. On construction of Hamilton cycles in graphs of distributed computer systems by a recurrent neural network // Proc. Xth Russian Scientific Conf. “Neuroinformatika-2008”, Moscow: Part 2. — 2008. — P. 76–85 (In Russian).

[13] Bianchi L., Knowles J., Bowler J. Local search for the probabilistic traveling salesman problem: Correction to the 2-opt and 1-shift algorithms // Eur. J. Oper. Res. — 2005. — Vol. 162. — No. 1. — P. 206–219.

[14] Reinelt G. TSPLIB — a traveling salesman problem library // ORSA J. Comput. — 1991. — Vol. 3. — No. 4. — P. 376–384.

[15] Neurocomputers in Spacecraft Engineering / V.V. Efimov, ed. // Scientific series “Neurocomputers and Their Application”. — Vol. 17. — Moscow: Radiotekhnika, 2004 (In Russian).