Finding Approximate Solutions to NP-Hard Problems by Neural Networks Is Hard[*][†]

Xin Yao[‡]
Computer Sciences Laboratory
Research School of Physical Sciences and Engineering
The Australian National University
GPO Box 4, Canberra, ACT 2601, Australia

Abstract

Finding approximate solutions to hard combinatorial optimization problems by neural networks is a very attractive prospect. Many empirical studies have been done in the area. However, recent research about a neural network model indicates that for any NP-hard problem the existance of a polynomial size network that solves it implies that NP=co-NP, which is contrary to the well-known conjecture that NP$\neq$co-NP. This paper shows that even finding approximate solutions with guaranteed performance to some NP-hard problems by a polynomial size network is also impossible unless NP=co-NP.

Keywords — Neural Networks, Combinatorial Optimization, Computational Complexity.

1 Introduction

The interest in mapping combinatorial optimization problems onto neural networks has been growing rapidly since Hopfield and Tank first used them to solve TSP [1]. It has been demonstrated that neural network optimization algorithms can give good near optimal solutions to rather large NP-hard problems [2]. However, recent work by Bruck and Goodman [3] indicates that under a simple neural network model for any NP-hard problem the existance of a polynomial size network that solves it implies that NP=co-NP, which is contrary to the well-known conjecture that NP$\neq$co-NP [4]. This paper extends the above result and shows that even finding approximate solutions with guaranteed performance to some NP-hard problems is also impossible unless NP=co-NP. The results given here can

*In **Information Processing Letters, 41**:93–98, 1992

†A preliminary version was presented at the 1991 International Conference on Artificial Neural Networks in Finland, 24-28 June 1991.

‡The author is now with the Commonwealth Scientific and Industrial Research Organization, Division of Building, Construction and Engineering, P.O.Box 56, Graham Road, Highett, Victoria 3190, Australia.

help us better understand what we could expect from a neural network for solving NP-hard problems, even though we only need approximate solutions. We will introduce the neural network model in Section 2 and some reliminaries in Section 3. The main results are shown in Section 4. Finally, some concluding remarks are made in Section 5.

2 The Neural Network Model

The neural network model used here is the same as that Hopfield and Tank first considered in [5]. It is a discrete time system which can be described as a weighted and completely connected undirected graph. Denote the neural network with n neurons (nodes) as $\mathcal{N}$. It can be uniquely defined by weights and thresholds (W, U), where $W = (w_{ij})_{n\times n}$ is a symmetric matrix, w_{ij} is the weight between neuron i and neuron j, and $U = (u_i)_n^{\mathrm{T}}$ is a column vector, u_i is the threshold of neuron i.

The state of the network at time t is represented by column vector $S(t) = (s_i(t))_n^{\mathrm{T}}$, where $s_i \in \{-1, +1\}$ is the state of neuron i at time t. The next state of neuron i is computed by threshold logic function

$$s_i(t+1) = sign(H(i,t)) = \begin{cases} +1 & if\ H(i,t) \geq 0 \\ -1 & otherwise \end{cases}$$

where

$$H(i,t) = \sum_{j=1}^{n} w_{ji} s_j(t) - u_i.$$

The energy function associated with network $\mathcal{N}$ is defined by [5]

$$E(t) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1,\, j\neq i}^{n} w_{ij} s_i(t) s_j(t) + \sum_{i=1}^{n} u_i s_i(t).$$

It has been shown that this neural network model always converges to a stable state when operating in a serial mode [5], thus can often be used in associated memories and combinatorial optimizations.

3 Preliminaries

In order to facilitate the following discussions, we first give some definitions about combinatorial optimzation [4].

Definition 3.1 *A combinatorial optimization problem* Π *is either a minimization problem or a maximization problem and consists of the following three parts:*

(1) a set D_Π *of instances;*

(2) for each instance $I \in D_\Pi$*, a finite set* $S_\Pi(I)$ *of candidate solutions for* I*; and*

(3) a function m_Π that assigns to each instance $I \in D_\Pi$ and each candidate solution $\sigma \in S_\Pi(I)$ a positive rational number $m_\Pi(I,\sigma)$, called the solution value for σ.

If Π is a minimization (maximization) problem, then an optimal solution for an instance $I \in D_\Pi$ is a candidate solution $\sigma^* \in S_\Pi(I)$ such that, $\forall \sigma \in S_\Pi(I)$, $m_\Pi(I,\sigma^*) \leq m_\Pi(I,\sigma)$ ($m_\Pi(I,\sigma^*) \geq m_\Pi(I,\sigma)$). We will use $OPT_\Pi(I)$ to denote the value $m_\Pi(I,\sigma^*)$ of an optimal solution for I.

Definition 3.2 *An algorithm $\mathcal{A}$ is an approximate algorithm for Π if, given any instance $I \in D_\Pi$, it finds a candidate solution $\sigma \in S_\Pi(I)$. The value $m_\Pi(I,\sigma)$ of the candidate solution σ found by $\mathcal{A}$ when applied to I will be denoted by $\mathcal{A}_\Pi(I)$. If $\mathcal{A}_\Pi(I) = OPT_\Pi(I)$ for all $I \in D_\Pi$, then $\mathcal{A}$ is called an optimization algorithm for Π.*

Definition 3.3 *The size of an instance I of a problem Π, denoted by $|I|$, is defined by the number of symbols in the description of I obtained from an encoding scheme for Π (e.g., a binary encoding scheme).*

As for the neural network model and the neural network optimization algorithm, we have the following definitions.

Definition 3.4 *The size of a neural network $\mathcal{N}$, denoted by $|\mathcal{N}|$, is defined by the number of bits needed to describe weights W and thresholds U.*

Definition 3.5 *Let Π be an optimization problem. Then $\mathcal{A}_\Pi$ is defined as the algorithm (if one exists) which given an instance $I \in \Pi$ generates the description for a neural network $\mathcal{N}_I$ in time polynomial in $|I|$, where $\mathcal{N}_I$ satisfies: (a) $|\mathcal{N}_I|$ is bounded by some polynomial in $|I|$; and (b) every local minimum of the energy function associated with $\mathcal{N}_I$ corresponds to an global optimum of I, and there is a polynomial time algorithm that given a local minimum will find the corresponding global optimum solution to I.*

Definition 3.6 *Let Π be an optimization problem. Then $\mathcal{A}_{\Pi_{appr}}$ is defined as the algorithm which given an instance $I \in \Pi$ generates the description for a neural network $\mathcal{N}_{I_{appr}}$ in time polynomial in $|I|$, where $\mathcal{N}_{I_{appr}}$ satisfies: (a) $|\mathcal{N}_{I_{appr}}|$ is bounded by some polynomial in $|I|$; and (b) every local minimum of the energy function associated with $\mathcal{N}_{I_{appr}}$ corresponds to a candidate solution to I, and there is a polynomial time algorithm that given a local minimum will find the corresponding candidate solution to I.*

Definition 3.7 *A neural network for finding a global optimum (approximate) solution efficiently to an optimization problem Π exists if the algorithm $\mathcal{A}_\Pi$ ($\mathcal{A}_{\Pi_{appr}}$) exists.*

It is clear that 'efficiently' here means solving an optimization problem by a polynomial size network and in a polynomial transformation time. However, no restrictions are set on the time for the network to converge. Thus, we actually relax the conditions of what 'efficiently' should mean. Under the above assumptions, Bruck and Goodman showed the following results [3].

Lemma 3.1 *Let Π be an NP-hard problem. Then the existance of a neural network for solving Π implies that NP=co-NP.*

Lemma 3.2 *Let $\varepsilon \geq 0$ be some fixed number. The existance of a neural network for finding an ε-approximate solution to TSP implies that P=NP.*

4 Finding Approximate Solutions to NP-hard Problems by Neural Networks

The power of neural networks has been a widely interested topic in researches. Lemma 3.1 has shown that neural networks cannot solve NP-hard problems efficiently unless NP=co-NP. This fact forces us to turn to other alternatives. An obvious one is to find an approximate solution, instead of the exact one, to an NP-hard problem. Although Lemma 3.2 indicates that finding an ε-approximate solution to TSP by neural networks is also inefficient, there is not any other result like Lemma 3.1 which gives a general conclusion about approximate solutions with guaranteed performance to NP-hard problems. In fact, as indicated by Papadimitriou and Steiglitz [6], it is very difficult to develop a similar theory in the approximate algorithm theory parallel to the NP-complete theory. One important reason is that the cost function is usually very sensitive to even a simple reduction. Hence, only one result about finding an approximate solution to TSP by neural networks cannot fully describe the characteristics of neural networks. More results are needed to better understand the power of the neural networks and relations between its computation time and its size.

The first problem we consider here is Minimum Set Cover (MSC): Given a collection C of subsets of a finite set S, find a subset $C' \subseteq C$ such that $\bigcup_{c \in C'} c = S$ and such that $|C'|$ is as small as possible.

Proposition 4.1 *For any given positive constant K, no neural network $\mathcal{N}_{I_{appr}}$ for finding an approximate solution efficiently to $I \in MSC$ with performance guaranteed by*

$$\left| V_{OPT_{MSC}(I)} - V_{\mathcal{N}_{I_{appr}}} \right| \leq K \tag{1}$$

exists unless NP=co-NP, where $V_{\mathcal{N}_{I_{appr}}}$ stands for the approximate solution to instance $I \in MSC$ and $V_{OPT_{MSC}(I)}$ the global optimum solution.

<u>*Proof:*</u> Suppose there exists such a neural network for finding an approximate solution efficiently to MSC with guaranteed performance, i.e., for all $I \in MSC$, there is an $\mathcal{N}_{I_{appr}}$ and it satisfies (1). We can show that such a neural network solves MSC efficiently and exactly, which is contrary to Lemma 3.1.

For any instance $I \in MSC$, which is a collection of subsets of a finite set S, we construct a new instance I', which consists of $K+1$ distinct copies of I and is a collection of subsets of a finite set S' consisting of $K+1$ distinct copies of S. Obviously, $V_{OPT_{MSC}(I')} = (K+1)V_{OPT_{MSC}(I)}$, and different copies of I cover different copies of S. Hence, a solution to I can be obtained just by selecting the copy with minimum cover. Let $V_{I_{min}}$ be the value of the solution. Then $V_{\mathcal{N}_{I'_{appr}}} \geq (K+1)V_{I_{min}}$. Thus,

$$(K+1)\left|V_{I_{min}} - V_{OPT_{MSC}(I)}\right| \leq \left|V_{\mathcal{N}_{I'_{appr}}} - V_{OPT_{MSC}(I')}\right| \leq K$$

That is, $\left|V_{I_{min}} - V_{OPT_{MSC}(I)}\right| = 0$. So, $\mathcal{N}_{I_{appr}}$ has found the exact optimum solution to I which can be any instance in MSC.

Q.E.D.

Similar results can also be obtained for other NP-hard problems using the same proving method, i.e., constructing a new instance from the original one by multiplication.

Proposition 4.2 *The result of Proposition 4.1 is also true if MSC is replaced by the Maximum Independent Set, Minimum Vertex Cover, Maximum Clique, Maximum Set Packing and Knapsack problems.*

<u>*Proof:*</u> Similar to the proof for Proposition 4.1.

Q.E.D.

The performance guarantee defined by the difference by (1) is a very strong requirement. A weaker measurement can be defined by the ratio between the approximate solution and the optimum solution.

$$R_{\mathcal{N}}(I) = \begin{cases} V_{\mathcal{N}_{I_{appr}}} / V_{OPT(I)}, & if\ minimization \\ V_{OPT(I)} / V_{\mathcal{N}_{I_{appr}}}, & if\ maximization \end{cases} \tag{2}$$

$$R_{\mathcal{N}} = \inf\{r \geq 1 \mid R_{\mathcal{N}}(I) \leq r\ \forall I \in D_{\Pi}\} \tag{3}$$

$$R_{\mathcal{N}}^{\infty} = \inf\{r \geq 1 \mid \exists N \in Z^{+},\ R_{\mathcal{N}}(I) \leq r\, \forall I \in D_{\Pi}\ satisfying\ V_{OPT(I)} \geq N\} \tag{4}$$

Obviously, a ratio close to 1 represents a good performance. Under such measurements, we have the following result.

Proposition 4.3 *Let Π be a minimization problem having all solution values in Z^{+}, and suppose that for some fixed $K \in Z^{+}$ the decision problem "Given $I \in D_{\Pi}$, is $OPT(I) \leq K$" is NP-hard. Then there does not exist neural networks $\mathcal{N}_{I_{appr}}\, \forall I \in D_{\Pi}$ such that $R_{\mathcal{N}} < 1 + 1/K$ unless NP=co-NP.*

<u>*Proof:*</u> Suppose there exist neural networks $\mathcal{N}_{I_{appr}}\, \forall I \in D_{\Pi}$ such that $R_{\mathcal{N}} < 1 + 1/K$, then for any $I \in D_{\Pi}$ we can find an approximate solution $V_{\mathcal{N}_{I_{appr}}}$ efficiently by a neural network, which satisfies

$$\frac{V_{\mathcal{N}_{I_{appr}}}}{V_{OPT(I)}} < 1 + \frac{1}{K}$$

It is apparent that $V_{OPT(I)} \leq K$ if and only if $V_{\mathcal{N}_{I_{appr}}} < K+1$, which can be decided in constant time after $V_{\mathcal{N}_{I_{appr}}}$ is obtained. Hence, the NP-hard decision problem "Given

$I \in D_\Pi$, is $OPT(I) \leq K$" can be solved by the neural network efficiently. This is contrary to Lemma 3.1 unless NP=co-NP.

Q.E.D.

The above result relates the absolute performance ratio (3) for finding approximate solutions to a class of optimization problems by neural networks. Some results about asymptotic performance ratio (4) can also be shown.

Proposition 4.4 *There does not exist a neural network which can find an approximate solution efficiently with $R^\infty_\mathcal{N} < 2$ to the Minimum Graph Colouring (MGC) problem unless NP=co-NP.*

Proof: Suppose there exists a neural network which finds an approximate solution efficiently with $R^\infty_\mathcal{N} < 2$ to MGC, then we can show that the NP-hard Graph 3-Colourability (G3C) problem can be solved by the network efficiently and exactly, which is contrary to Lemma 3.1 unless NP=co-NP. The method used here follows [7, 4].

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. Define $G = G_1[G_2]$ as the composition of the two graphs, where

$$\begin{aligned} V &= V_1 \times V_2 \\ E &= \{\{(u_1,u_2),(v_1,v_2)\} \mid \text{ either } \{u_1,v_1\} \in E_1 \text{ or } u_1 = v_1 \text{ and } \{u_2,v_2\} \in E_2\} \end{aligned}$$

The graph $H_n = (V_n, E_n)$, $n \geq 6$ is defined as follows: It has $n(n-1)(n-2)/6$ vertices, each corresponding to a distinct 3 element subset of $\{1, 2, \cdots, n\}$, and two such vertices are joined by an edge in E_n iff their corresponding subsets have no element in common.

Define a (k,m)-multicolouring of a graph $G = (V,E)$ as a function f which assigns to each $v \in V$ a set $f(v) \subseteq \{1, 2, \cdots, k\}$ such that $|f(v)| = m$ for all vertices v and such that, whenever $\{u,v\} \in E$, then $f(u)$ and $f(v)$ contain no elements in common. The m-chromatic number $\chi_m(G)$ is the least integer $k \geq 1$ such that there exists a (k,m)-multicolouring of G. It has been shown that [7], for $n \geq 6$

$$\chi_3(H_n) = n \quad (5)$$

$$\chi_4(H_n) = 2n - 4 \quad (6)$$

Since $R^\infty_\mathcal{N} < 2$, there exists a neural network $\mathcal{N}$ and an $\varepsilon > 0$ and a $K \in Z^+$ such that $V_{\mathcal{N}_{I_{appr}}} \leq (2-\varepsilon)V_{OPT(I)}$ for all $I \in MGC$ with $V_{OPT(I)} \geq K$. Let $N \geq \max\{6, K\}$ be an integer chosen so that $2N - 4 > (2-\varepsilon)N$.

For any instance $I = (V,E) \in G3C$, we first construct $I^* = H_N[I]$, which can be done in time polynomial in $|I|$. Then we generate, in time polynomial in $|I^*|$ (also in $|I|$), a description of the neural network $\mathcal{N}_{I^*_{appr}}$, which finds an approximate solution $V_{\mathcal{N}_{I^*_{appr}}}$ to I^* efficiently. We will show that I is 3-colourable iff $V_{\mathcal{N}_{I^*_{appr}}} < 2N - 4$.

If I is 3-colourable, then I^* can be coloured by the least number of colours, N, because of (5). That is, $V_{\mathcal{N}_{I^*_{appr}}} \leq (2-\varepsilon)V_{OPT(I^*)} \leq (2-\varepsilon)N < 2N-4$. If I is not 3-colourable, then, by the definition of I^*, any colouring of I^* must use at least four distinct colours on each copy of I^*. Hence, from (6) we get $V_{\mathcal{N}_{I^*_{appr}}} \geq V_{OPT(I^*)} \geq 2N - 4$. Therefore, I is 3-colourable iff $V_{\mathcal{N}_{I^*_{appr}}} < 2N - 4$, which can be solved by the neural network $\mathcal{N}_{I^*_{appr}}$ efficiently and exactly. This is contrary to Lemma 3.1 unless NP=co-NP.

Q.E.D.

5 Concluding Remarks

Although the paper has shown some results in neural networks parallel to the classical NP-complete theory [4], it is unclear if we can develop a whole parallel theory, especially on approximate solutions to NP-hard problems. It is very important to study whether an NP-hard problem which can be solved approximately can also be solved by a neural network approximately under the same performance requirement, and whether other neural network models, e.g., the asymmetric model and the continuous model, are more powerful than that considered here.

Simon has recently demonstrated a kind of approximability-preserving transformations, called continuous reductions, among combinatorial optimization problems [8]. The performance ratio retains the same (up to a bounded factor) for two problems whenever they are mutually related by continuous reductions. Therefore, lower and upper bounds or gap-theorems valid for a particular problem can be transferred along reduction chains. For these problems, if one of them cannot be solved be neural networks efficiently with some performance ratio, then neither can all of them with the corresponding performance ratios.

In essence, performance guarantees are worst-case bounds, and approximate algorithms often behave much better in practice than their worst cases. Hence, a more realistic way to analyze neural networks is to investigate their average-case performance. Unfortunately, such analyses are usually very complicated in theory, and they have to suppose the instance distribution of a problem, which is difficult to know. An alternative is to study the behaviour of a neural network model empirically. Undoubtedly, further research into the behaviour of neural networks for finding approximate solutions to optimization problems is very important.

Acknowledgement — The author would like to thank Prof. R. P. Brent and Dr. I. D. G. Macleod for their support of the work.

References

[1] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, Vol. 52, pp.141-152, 1985.

[2] C. Peterson, "Parallel distributed approach to combinatorial optimization: benchmark studies on Traveling Salesman Problem," *Neural Computation*, Vol. 2, pp.261-269, 1990.

[3] J. Bruck and J. W. Goodman, "On the power of neural networks for solving hard problems," *Journal of Complexity*, Vol. 6, pp.129-135, 1990.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman Co., San Francisco, 1979.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, Vol. 79, pp.2554-2558, 1982.

[6] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization Algorithms and Complexity*, Prentice-Hall Inc., 1982.

[7] M. R. Garey and D. S. Johnson, "The complexity of near-optimal graph coloring," *J. Assoc. Comput. Mach.*, Vol. 23, pp.43-49, 1976.

[8] H. U. Simon, "On approximate solutions for combinatorial optimization problems," *SIAM J. Disc. Math.*, Vol. 3, pp.294-310, 1990.