A Pointer Network Based Deep Learning Algorithm for the Max-Cut Problem

Shenshen Gu(✉) and Yue Yang

School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China
gushenshen@shu.edu.cn

Abstract. The max-cut problem is one of the classic NP-hard combinatorial optimization problems. In order to solve this problem efficiently, the paper mainly studies the topic of using the pointer network to build a training model to solve the max-cut problem. Then, the network model is trained with supervised learning. The experimental results show that the network trained by this algorithm can obtain the approximate solution to the max-cut problem.

Keywords: Max-cut problem · Pointer network · Supervised learning

1 Introduction

The max-cut problem belongs to the famous twenty-one NP (Nondeterministic Polynomial) problems that Richard M. Karp first proposed [1]. The max-cut problem refers to finding a maximum segmentation for a given directional weighted graph that maximizes the total weights across all edges of these two cut sets [2].

As a typical NP hard problem in combinatorial optimization problem, the max-cut problem has various applications in statistical physics, image processing, communication network design, circuit layout design and other engineering problems. In view of the dual important value of the theory and practice, in the past few decades, researchers have proposed various algorithms to solve the max-cut problem. The algorithm can be divided into two categories, one of which are exact algorithms and the other are heuristic algorithms. The exact algorithms include the enumeration method [3] and the branch and bound method [4] etc. Although the optimal solution to this problem can theoretically be found by an exact algorithm, it is often impossible to achieve it, because the computational time increases exponentially with the increase of the scale of the problem, the search space for the problem also increases rapidly as the scale increases. Even if the current state-of-the-art computer is used for calculation, the time for solving the problem is not tolerable. Therefore, finding an effective approximate heuristic algorithm is of great significance. The effective methods for solving the max-cut problem include immune algorithm, genetic algorithm, greedy algorithm, ant colony algorithm, simulated annealing algorithm, LKH algorithm [5],

© Springer Nature Switzerland AG 2018
L. Cheng et al. (Eds.): ICONIP 2018, LNCS 11301, pp. 238–248, 2018.
https://doi.org/10.1007/978-3-030-04167-0_22

etc. Compared with the exact algorithms, the heuristic algorithms can be applied to solving large-scale problems with thousands or even tens of thousands of variables in a short period of time, so computational efficiency is improved. However, there is a defect that cannot be ignored in the heuristic algorithms, that is, the degree of deviation between the feasible solution and the optimal solution cannot be accurately predicted all the time, and it is easy to fall into a local optimal solution. Therefore, it is of great theoretical significance and application value to study the effective algorithms for solving the max-cut problem.

Recently, deep learning based methods are becoming more and more popular due to the fact that they are capable of discovering their own heuristics based on abundant training data automatically. For this reason, except for the famous application in computer vision, image classification [6] and speech recognition [7], deep learning based methods are now making potential progress in solving various combinatorial optimization problems. Deep learning can represent the categories or features of the data by extracting the underlying features of the combined optimization problem data to form more abstract high-level features, and then using the distributed features of the data. For instance, the famous TSP problem is successfully solved by Oriol Vinyals with RNNs [8]. The quadratic assignment problem is effectively solved by Anto Milan with a data-driven approach [9]. Inspired by these important ideas, a deep learning based method to solve the max-cut problem is proposed in the paper.

The rest of this paper is organized as follows. Section 2 introduces the formulation of the max-cut problem and the architecture of the pointer network. Section 3 explains how to use the pointer network to solve the max-cut problem. Then, Sect. 4 details the experiments and analysis. And finally, the conclusion is given in Sect. 5.

2 Problem Formulation

In this section, the mathematical description of the max-cut problem is first introduced and then the architecture of the pointer network model is described.

2.1 The Max-Cut Problem

$G = (V, E)$ is a graph, where $V = \{1, 2, \ldots n\}$ is vertex set and E is edge set. Suppose that w_{ij} is the weight for edge (i, j) in E. Dividing the vertex set V into two subsets S and S', satisfying $S \cup S' = V$ and $S \cap S' = \varnothing$, then calling S and S' constitute a cut of the graph G. The value of the cut is the number of edges with one end in S and the other end in S', it is calculated by the following equation:

$$cut\,(S, S') = \sum_{\substack{u \in S \\ v \in S'}} w_{uv} \tag{1}$$

The max-cut problem consists of finding a cut in G with maximum value.

In this paper, we assume that the weight on each edge is one without loss of any generality. Our goal is to find a segment (S, S') of the vertex set V, so that the maximum number of edges is divided (i.e., one vertex of the edge in S and the other vertex in S').

2.2 The Pointer Network

Pointer network is a new type of deep neural architecture combines the popular sequence-to-sequence learning framework [10] with a modified attention mechanism [11] to learn the conditional probability of an output whose values correspond to positions in a given input sequence. It was first proposed by Vinyals et al. [8] to solve TSP problems. The neural network architecture for solving the max-cut problems is shown in Fig. 1. The structure of the pointer network is briefly introduced as follows

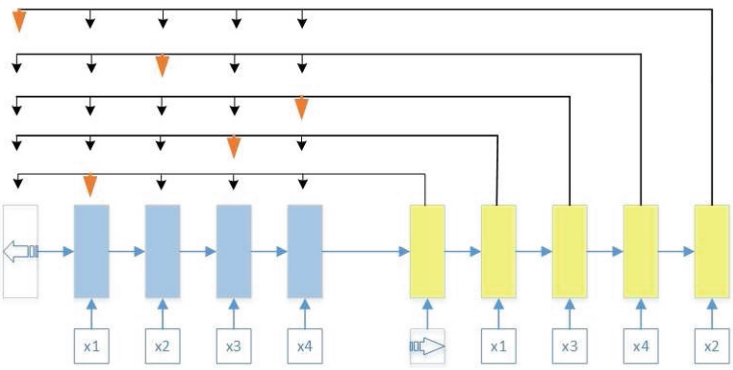

Fig. 1. Architecture of the pointer network (encoder in blue, decoder in yellow) (Color figure online)

The Seq2seq module is mainly composed of an encoder and a decoder. The encoder represents a variable-length input sequence as a vector of fixed dimensions, and the decoder converts this vector into a variable-length output vector. The attention mechanism that connects the encoder and the decoder allows the decoder to query the entire sequence of encoder states, not just the last LSTM cell state. The attention mechanism is actually using a variable-length vector to extract relevant information from the input. It generates corresponding weights for each element of the input sequence, indicating the degree of correlation with the next input of the decoding section. It purposed to tell the decoder network which input parts are more important. This method allows the decoder to focus more on finding useful information in the encoder input sequence that is relevant to the current output, thereby improving the quality of the output.

In the model of this paper, RNN networks are constructed with LSTM units. LSTM is a special recurrent neural network architecture. Compared with feed-forward neural networks, RNN has the characteristics of cyclic connections, making it more suitable for the modeling of sequences [12]. The sequence X is fed to the decoder and one element is fed into each time step until the end of the sequence. The end of the sequence is marked with a special end marker. The model then switches to decoding mode, where each time step produces an element in the output sequence of the decoder until the end marker appears. Until this time, the entire process ended.

Each conditional probability of encoder and decoder can be defined as

$$\begin{aligned} p\left(y_i \left| y_1, \ldots, y_{i-1}, X\right.\right) &= g\left(y_{i-1}, d_i, c_i\right) \\ d_i &= h\left(d_{i-1}, y_{i-1}, c_i\right) \end{aligned} \tag{2}$$

The c_i vector is calculated as follows:

$$c_i = \sum_{j=1}^{n} \alpha_j^i e_j \tag{3}$$

Where d_i and e_j in (2) and (3) are the hidden states of the decoder and the encoder, respectively, and the weights α_j^i are defined as:

$$\begin{aligned} \alpha_j^i &= \frac{\exp\left(u_j^i\right)}{\sum_{k=1}^{n} \exp\left(u_k^i\right)} \\ \left(u_j^i\right) &= a\left(d_{i-1}, e_j\right) \end{aligned} \tag{4}$$

Among them, a is a feed forward neural network, and the vector u_j^i is called the attention mark of the input sequence element.

Prior to the introduction of the pointer network, there is a problem with the model of Seq2seq combined with the attention mechanism, that is, the output dictionary size of the encoder must depend on the length of the input sequence. Therefore, the pointer network is used to adjust the standard attention mechanism and create a pointer u_j^i to the input sequence element, so that the extra information propagated to the decoder is no longer just the final state of the encoder [13]. Instead, using u_j^i to point to the input sequence element.

$$\begin{aligned} u_j^i &= v^T \tanh\left(W_1 e_j + W_2 d_i\right) \quad \mathrm{j} \in (1, \ldots, n) \\ p\left(C_i \left| C_1, \ldots, C_{i-1}, P\right.\right) &= soft\max\left(u^i\right) \end{aligned} \tag{5}$$

Where $softmax$ normalizes vector u_j^i of length n to make it an output probability distribution on the input dictionary, and v, W_1, W_2 are the parameters that can be learned in the model, and $C = C_1, \ldots, C_m$ is a sequence of m indices.

3 Solving the Max-Cut Problem Using the Pointer Network

3.1 Data Structure of the Max-Cut Problem

In the max-cut problem, our goal is to find a point set S that can make the $cut\,(S, S')$, which is the sum of the weights on the edges in E, obtain the maximum value.

Inspired by Vinyals' idea of solving the TSP problem that uses the trained neural network model, input the set of city node coordinates and output the predicted probability distribution of the various nodes of these cities. For the max-cut problem, the input to this network is the weight of the line between the point and the point. The input is represented by a matrix, i.e. w_{ij} represents the weight of the line between point i and point j ($w_{ij} = 1$ or 0, where 1 means there exist a connection between two points and 0 means there is no connection between two points). The output of the network is the segmentation of the vertex set V. The output represents all points in order by 0 and 1, where points marked "1" are placed in one set and points marked "0" are placed in the other set.

For example, let $G = (V, E)$ be an undirected graph with seven vertices. And the weight on edge (i, j) are set to w_{ij} ($w_{ij} = w_{ji}$). This problem can be written as

$$\begin{aligned} &f(x) = x_1x_2 + x_1x_5 + x_2x_5 + x_2x_7 + x_4x_5 + x_4x_7 \\ &x_i \in \{0, 1\}, (i = 1, \ldots, 7) \end{aligned} \tag{6}$$

The input weight matrix W is

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

The problem's optimal solution is $x = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}^T$. That means vertices x_3, x_4, x_6, and x_7 belong to one set, while vertices x_1, x_2, and x_5 belong to the other set.

3.2 Datasets Generation

Our experiments use the MATLAB program to randomly generate 100 sets of samples as a training set. Each time, ten sets of samples are extracted for training, and 100 sets of samples are generated as a test set.

3.3 Supervised Learning

Supervised learning is a method often used in machine learning. It can learn or create a learning model through training data, and infers the output corresponding to the new given instance input based on this model. The training data consists of the input (usually a vector) of the corresponding problem and the corresponding expected output. The output of a function can be a continuous value (called a regression analysis) or it can be a classification label of a prediction (called a classification). A task of supervised learning is to predict the output of the function corresponding to any possible input value after observing some typical training (input and corresponding expected output). In order to achieve this goal, learners must generalize from existing data to non-observed situations in a "reasonable" manner. This situation is commonly referred to as concept learning in human and animal perceptions.

Supervised learning is, in simple terms, a classification that people often say. Through the existing training samples, which are known data and their corresponding outputs, they are trained to obtain an optimal model. This model is then used to map all inputs to the corresponding outputs and make simple judgments on the outputs to achieve correct classification. So the model has the ability to judge and classify unknown data.

In order to solve a given problem of supervised learning (such as the max-cut problem), the following steps must be considered.

– Initializing Network

Set up hyperparameters, for instance, the number of layers in the neural network, learning rate, the type of neuron activation function, batch size and the method of weight initialization.

– Loading Data

Determine the representation of the input features of the learning function. Convert the input and output data formats to the desired data format.

– Producing a Network Model

Creat a sequence model, attention mechanism functions, loss functions, and optimization functions.

– Training Network

Train the network model and adjust parameters.

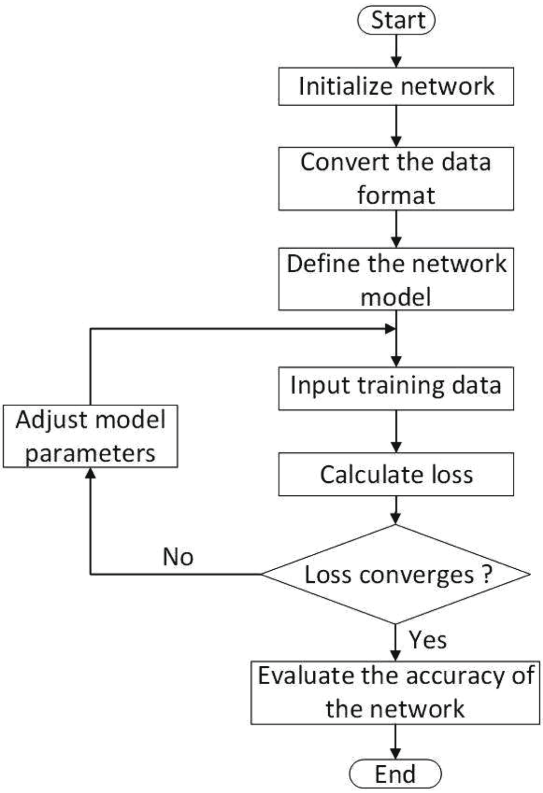

Fig. 2. Scheme of neural network

– Evaluating Network

Use the test set to assess the accuracy of the network on solving the max-cut problem.

The total procedure of the network model is shown in Fig. 2.

4 Experiments Results and Analysis

The pointer network for solving the max-cut problem was implemented with TensorFlow.

In order to validate the pointer network, we performed five experiments. In these experiments, five data sets were generated randomly, with dimensions of 10, 20, 30, 40 and 50. (the dimensions are the number of vertices). Taking the 20-dimensional (see Fig. 3) max-cut problem as an example. The result of 20-dimensional with 1000 training times and 100 training samples is given as follows.

```
Total training time:  0:09:45.783529
Predicted solution:  [ 1. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 0. 1. 0. 0. ]
Predicted solution:  [ 1. 0. 1. 1. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. ]
Predicted solution:  [ 1. 1. 0. 1. 0. 1. 0. 0. 0. 1. 1. 0. 1. 1. 0. 1. 0. 1. 0. 0. ]
Predicted solution:  [ 1. 1. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0. 0. 1. ]
Predicted solution:  [ 1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0. 1. 1. 1. ]
Predicted solution:  [ 1. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 0. ]
Predicted solution:  [ 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 0. 0. ]
Predicted solution:  [ 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 0. 1. 0. ]
Predicted solution:  [ 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. ]
Predicted solution:  [ 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. ]
Optimal value:
 [ 71.0, 74.0, 74.0, 74.0, 74.0, 72.0, 75.0, 74.0, 76.0, 72.0 ]
Predicted value:
 [ 61.0, 66.0, 67.0, 66.0, 66.0, 64.0, 66.0, 66.0, 72.0, 70.0 ]
Accuracy:
 [ 0.859  0.892  0.905  0.892  0.892  0.889  0.88  0.892  0.947  0.972 ]
Accuracy of sum:
 0.902
```

Fig. 3. Experimental results of 20-dimensional max-cut problem

Figure 3 shows the training time, the predicted solution to the tested max-cut problem, the optimal value and the predicted value, the accuracy of the network for the ten groups on 20-dimensional max-cut problem. The results in Tables 1 and 2 below were obtained in the same manner as Fig. 3.

As can be seen from the figure, by repeating the training of 10 sets of 20-dimensional data, we obtained the optimal solution to the expected output "Predicted solution", points with "1" means they were placed in one group, and points with "0" means they were placed in the other group. "Optimal value" represents the real optimal value of the input point set, and "Predicted value" represents the predicted optimal value obtained after the input point set was trained by the neural network. "Accuracy" is the ratio of "Predicted value" to "Optimal value", this parameter is used to indicate the quality of the trained model. "Accuracy of sum" is the average of 10 sets of "Accuracy".

Table 1 shows the training time of the max-cut problem model on five different dimensions, t indicates the times of training and s indicates the number of training samples.

Table 2 shows the solution accuracy of the max-cut problem model on five different dimensions with different times of training (t) and the number of training samples (s).

Table 1. Training time of the max-cut problem model

Dimensions	10	20	30	40	50
t = 1000, s = 100	5'16"	9'46"	14'09"	18'42"	23'15"
t = 1000, s = 1000	5'13"	9'36"	14'06"	18'40"	23'13"
t = 2000, s = 100	10'29"	19'31"	28'19"	37'28"	46'21"
t = 2000, s = 1000	10'25"	19'13"	28'14"	37'28"	46'28"

Table 2. Accuracy of the max-cut problem model

Dimensions	10	20	30	40	50
t = 1000, s = 100	95.2%	90.2%	85.2%	81.0%	68.7%
t = 1000, s = 1000	92.0%	91.6%	86.4%	81.3%	79.0%
t = 2000, s = 100	97.5%	88.3%	85.7%	84.3%	73.7%
t = 2000, s = 1000	95.1%	94.2%	86.1%	85.2%	81.4%

Figure 4 and the above two tables show that with the progressive increase of the dimension of the max-cut problem, the time spent on training gradually increases, and the accuracy of the approximate solution decreases. Then, as the training times increase, the quality of the approximate solution is also improved. In addition, The larger the number of training samples, the higher the accuracy of the approximate solution.

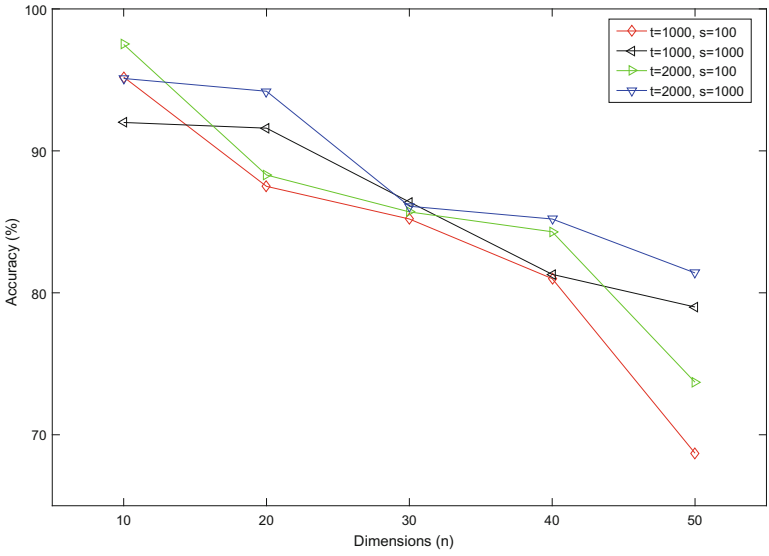

Fig. 4. The accuracy of five dimensions max-cut problem

5 Conclusion

In this paper, we use the pointer network to solve the max-cut problem. The proposed neural network architecture is a variant of the Seq2seq model, which can utilize RNN ordered connections to convey information and allow information to be persisted to predict the final solution. Experiments of the max-cut problem with different dimensions demonstrate that the supervised learning based method can obtain a nice approximate solution. This method greatly reduces the time and cost of calculations compared to conventional algorithms. The experimental results can be said to be very satisfactory, and some of the experimental results even reached the optimal value. The results obtained from the five sets of experiments allow us to see the advantages of solving the combinatorial optimization problem using a pointer network. It indicates that the method has great application potential in exploring combinatorial optimization problems.

Acknowledgments. The work described in the paper was supported by the National Science Foundation of China under Grant 61876105.

References

1. Mehlhorn, K.: NP-completeness. Eatcs Monogr. Theor. Comput. Sci. **5**(3), 359–376 (1984)
2. Bie, T.D., Cristianini, N.: Fast SDP relaxations of graph cut clustering, transduction, and other combinatorial problem. J. Mach. Learn. Res. **7**(3), 1409–1436 (2006)
3. Croce, F.D., Kaminski, M.J., Paschos, V.T.: An exact algorithm for MAX-CUT in sparse graphs. Oper. Res. Lett. **35**(3), 403–408 (2007)
4. Krishnan, K., Mitchell, J.E.: A semidefinite programming based polyhedral cut and price approach for the maxcut problem. Comput. Optim. Appl. **33**(1), 51–71 (2006)
5. Funabiki, N., Kitamichi, J., Nishikawa, S.: An evolutionary neural network algorithm for max cut problems. In: International Conference on Neural Networks, vol. 2, pp. 1260–1265. IEEE (1997)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, vol. 60, pp. 1097–1105. Curran Associates Inc. (2012)
7. Deng, L., Li, J., Huang, J.T., Yao, K., Yu, D., Seide, F., et al.: Recent advances in deep learning for speech research at Microsoft. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8604–8608. IEEE (2013)
8. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: International Conference on Neural Information Processing Systems. MIT Press (2015)
9. Milan, A., Rezatofighi, S.H., Garg, R., Dick, A., Reid, I.: Data-driven approximations to NP-hard problems (2017)
10. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks, vol. 4, pp. 3104–3112 (2014)
11. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. Comput. Sci. (2014)

12. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large vocabulary speech recognition. Comput. Sci. 338–342 (2014)
13. Acuna-Agost, R., Acuna-Agost, R.: Deep choice model using pointer networks for airline itinerary prediction. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1575–1583. ACM (2017)
14. Zhou, M.X.: A benchmark generator for Boolean quadratic programming. Comput. Sci. (2015)
15. Barahona, F., Junger, M., Reinelt, G.: Experiments in quadratic 0–1 programming. Math. Program. **44**(1–3), 127–137 (1989)
16. Gu, S., Hao, T.: A pointer network based deep learning algorithm for 0–1 Knapsack Problem. In: International Conference on Advanced Computational Intelligence (ICACI 2018), pp. 357–361 (2018)
17. Gu, S., Hao, T., Yang, S.: The implementation of a pointer network model for traveling salesman problem on a Xilinx PYNQ board. In: Huang, T., Lv, J., Sun, C., Tuzikov, A.V. (eds.) ISNN 2018. LNCS, vol. 10878, pp. 130–138. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92537-0_16