**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 3, No. 3, 2023

# One-shot Based Knowledge Graph Embedded Neural Architecture Search Algorithm

**Wenli Li, Gang Wu***

School of Computer and Engineering, Northeastern University, Shenyang 110819, China
* **Corresponding author**: Gang Wu (Email: wugang@mail.neu.edu.cn)

**Abstract:** The quality of embeddings is crucial for downstream tasks in knowledge graphs. Researchers usually introduce neural network architecture search into knowledge graph embedding for machine automatic construction of appropriate neural networks for each dataset. An existing approach is to divide the search space into macro search space and micro search space. The search strategy for micro space is based on one-shot weight sharing strategy, but it will lead to all the information obtained from the previous supernet training is discarded and the advantages of one-shot algorithm are not fully utilized. In this paper, we conduct experiments on common datasets for two important downstream tasks of knowledge graph embedding entity alignment and link prediction problems, respectively-and compare the search performance with existing manually designed neural networks as well as good neural network search algorithms. The results show that the improved algorithm can search better architectures for the same time when experiments are performed on the same dataset; the improved algorithm takes less time to search architectures with similar performance. Also, the improved algorithm searched the model on the dataset due to the human optimal level.

**Keywords:** One-shot algorithm; Weight sharing strategy; Knowledge graph embedding.

## 1. Introduction

Knowledge Graph (KG) is a method to represent and store things and their relationships through graph models. However, the information contained in a knowledge graph is often incomplete, and as knowledge accumulates, many new knowledges will be generated. At this time, the problems of difficult inference of the relationships between entities and severe data sparsity in knowledge graphs are constantly highlighted. This brings inconvenience to the further application of knowledge graphs in industrial production. In recent years, Knowledge Graph Embedding (KGE), which is the core technology of knowledge representation, has made new research breakthroughs, and the main technical implementation route of KGE is to use the translation invariance of word vectors to embed the high-dimensional entities and relations in the knowledge graph into the low-dimensional vector space, so that the semantic reasoning process of entities and relations can be converts the semantic inference process of entities and relations into the computation of distances between objects in the vector space.

The Interstellar [6] algorithm greatly improves the search efficiency of NAS and the performance of searching to architectures. However, the Interstellar algorithm initializes the microarchitecture parameter probability distribution matrix by taking the mean value when searching for the corresponding microarchitecture for each microarchitecture. This algorithm design makes it difficult to reuse the information related to the microarchitecture learned in each round. Considering that the microarchitecture is a supernet with shared parameters, a one-time evaluation strategy is used. Inspired by this, we consider using the probability distribution matrix of microarchitecture parameters updated in the previous round to initialize the information for the next round to better utilize the information from the previous round and improve the search efficiency.

In this paper, we first introduce the current research status of algorithm-related theories, next introduce the idea of search space partitioning, and provide a specific description of algorithm improvement, ideas and algorithm flow, and describe the analysis of experimental setup and experimental results. The performance of the improved algorithm is compared with the architecture searched by the original algorithm and the excellent architecture designed by human hand, and finally a summary and a discussion of future work are given.

## 2. Related Work

### 2.1. Knowledge Graph Embedding (KGE)

The main research objective of the KGE task is to map the entities and their relations in a knowledge graph representing a high-dimensional space one by one into a low-dimensional continuous vector space, which is also called Representation Learning in Knowledge (RLK).

A knowledge graph is composed of a triad representing entities and relationships. Therefore, KGE models can be divided into triplet-based models and path-based models. In terms of organization, knowledge graphs are complex network structures consisting of entities and relationships, and researchers have also proposed some graph-based models (GCN-based). Triplet-based models interpret the interactions between triads in a different way, path-based models can learn both relations and entities and relations on the path, and graph-based models can make full use of graph structure features to provide more accurate and explanatory semantic embeddings for entities and relations.

### 2.2. Neural Architecture Search (NAS)

With the success of deep learning, neural networks are used in more and more fields, different network architectures are developed and applied to different fields, neural networks become more and more complex, and the difficulty of designing neural networks manually is increasing. The

concept of Neural Architecture Search (NAS) was first introduced in 2017.NAS focuses on how to make machines automatically search for neural networks with better performance faster. Currently, neural networks built using NAS methods have outperformed neural networks designed by humans on these tasks for tasks such as image classification, target detection, and semantic segmentation.

There are three main research directions of NAS [2]: how to construct a complete and efficient search space, how to search neural networks quickly, and how to evaluate the performance of the searched neural networks more scientifically. The relationship between the search space, search strategy, and performance evaluation strategy is shown in Figure 1.

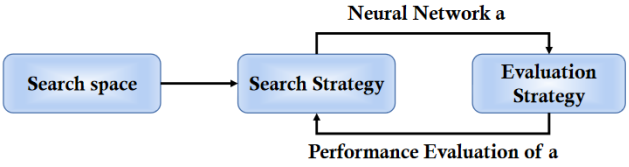


**Figure 1.** Example of a generic NAS algorithm idea

The search strategy selects one or several architectures from a predefined search space A. These architectures are passed to the performance evaluation module, which feeds the performance evaluation scores of these architectures to the search strategy to guide the optimization of the next search.

We first consider the search space, which should be complete so as to ensure that the search space contains the optimal architectures, but when the search space is too large, the search process consumes a lot of time and computational resources, and it is difficult to find the optimal architectures. Therefore, researchers generally tailor specific search spaces for specific tasks to improve the search space according to the characteristics of the tasks to be solved.

The search strategy and performance evaluation strategy are often closely related, with the search strategy describing the way we search the search space and the performance evaluation strategy describing the way we evaluate the performance of the searched architecture and guide the next architecture search process. Currently, there are two main mainstream approaches for NAS, a stand-alone evaluation strategy (Stand-alone Method) and a one-time evaluation strategy [3] (One-shot Method). The difference between them mainly lies in whether the architectural parameters are shared among different architectures in the search space. The independent evaluation strategy starts from the initial state to train and evaluate each model, which is a more reliable evaluation method because of the accurate evaluation results. The core idea of one-time evaluation lies in Parameter Share, which treats the search space as a supernet, and different neural networks as subnets on the supernet, and different subnets share the parameters of the supernet, this method avoids training each model from scratch and is more efficient, but the accuracy of evaluation decreases as the search space increases, therefore, the one-time evaluation strategy is mostly used on small search spaces.

## 3. Introduction of Preliminary Algorithms

The networks commonly used to solve the knowledge graph embedding problem are Graph Neural Network (GNN) and Recurrent Neural Network (RNN) [4]. Here we use Recurrent Neural Network, which is good at processing sequential data, as the target architecture. In this problem, the recurrent neural network obtains the embedding of $s1, r1$ by iteratively processing $sL, rL$. The input of the architecture at each step t is $s_t$,$r_t$, $h_{t-1}$, and the output is $v_t$,$h_t$. The whole architecture can be defined as the following recursive function.

$$[v_t, h_t] = f(s_t, r_t, h_{t-1}), \forall t = 1 \dots L \quad (1)$$

Where, $h_t$ is the implicit state in the architecture, initially $h_0 = s_1$. The output $v_t$ is used to predict the tail entity $o_t$.

In the search process, if all the parameters to be searched are put into one search space, the search space will be quite huge. If a stand-alone evaluation method is used to evaluate the architecture, it will give accurate feedback on the performance of the architecture, but it will take a long time and consume huge resources [5]. If a one-time evaluation method is used to evaluate the architectures, an accurate performance evaluation of the candidate architectures cannot be obtained because of the huge search space.The Interstellar algorithm [6] proposes to define all possible architectures as a search space and divide the entire search space into a macroscopic search space and a microscopic search space according to the degree of impact on information flow. The results of the delineation are shown in the following Table 1.

There are about $6 \times 10^5$ candidate architectures in the search space before the search space partition, which is quite large and the search process is extremely computationally intensive. There are $4^2 \times 4^3 = 1024$ architectures in the macro space and $3^2 \times 2^6 = 576$ architectures in the micro space after the partition. It can be seen that after the partitioning, both the macroscopic space and the microscopic space achieve an exponential decrease in size compared with the space before the partitioning. The macroscopic search space is larger and uses an independent evaluation strategy, while the microscopic space is smaller and uses a one-time evaluation strategy.

**Table 1.** Description of macrospace and microspace division

| macro-level | | micro-level | |
|---|---|---|---|
| **connections** | **combinators** | **activation** | **weight matrix** |
| $h_{t-1}, O_s, 0, s_t$ | +,Θ,⊗,gated | identity, tanh, sigmoid | $\{W_i\}_{i=1}^{6}.I$ |

## 4. Algorithm introduction

The current mainstream evaluation methods for neural network architectures are independent evaluation methods and one-time evaluation methods. The independent evaluation methods start from initialization to train each subnet and get their performance information, but this will bring huge time overhead and computational cost. To solve this problem, researchers have proposed the one-time evaluation method. The core idea of the one-time evaluation method is that the subnets share the parameters of the supernet and trade accuracy for speed. This strategy greatly reduces the time of the whole process and becomes the current

mainstream evaluation strategy.

In order to take full advantage of the one-shot algorithm in microspace, we make the following improvements to the Instellar algorithm. In the original algorithm, when searching for the corresponding micro-architecture for each macro-architecture, the micro-architecture parameter probability distribution matrix is initialized using the mean value method. In order that each search for microarchitecture can utilize the information from the previous round, we choose to initialize the matrix with the updated probability distribution matrix of microarchitecture parameters from the previous round for the next round. The flow of our algorithm is roughly.

**Algorithm**

1 Initialize the macro-architecture parameter probability distribution matrix
2 Sample the macro architecture
3 Initialize the micro-architecture parameter probability distribution matrix with the mean if it is the first round; if it is not the first round, initialize it with the micro-architecture probability distribution matrix obtained in the previous round.
4 Neural network training, update the micro-architecture probability distribution matrix by back propagation
5 Sampling micro-architecture
6 Update the macro-architecture parameter probability distribution matrix
7 Repeat steps 2-6 until a suitable architecture is searched

# 5. Experiment

## 5.1. Experimental Setup

We use a biased random walk approach to sample paths, using two basic tasks in knowledge mapping, entity alignment and link prediction, as applicable scenarios. Ranking-based MRR, Hit@1 and Hit@10, are used as task metrics. The experiments are run on a single GeForce RTX3090 64GB using the Pytorch framework [7].

## 5.2. Introduction to the experimental dataset

For the entity alignment task, we chose three cross-language and cross-database datasets, which are subsets of DBpedia and Wikidata, namely DBP-WD, EN-FR, and EN-DE. for each dataset, by divided into normal and dense versions. The normal version of the dataset samples the triples with degrees that approximate the nodes in the original knowledge graph. This sampling method makes the dataset more realistic and closer to the original knowledge graph. The dense version of the dataset randomly removes the entities with lower degrees from the original knowledge graph. This version of the dataset more closely resembles the dataset used by existing methods [8].

For the link prediction task, the commonly used datasets are WN18 and FB15, but they are prone to test leakage through inverse relations. That is, the test triples often contain many triples obtained by performing inverse operations on the triples in the training set. To solve the problem of link leakage, datasets WN18-RR and FB15K-237, which are subsets of datasets WN18 and FB15, have been created without inverse relations.

**Table 2.** Dataset

| Version | Data Source | DBP-WD | | EN-DE | | EN-FR | |
|---|---|---|---|---|---|---|---|
| | | DBpedia | Wikidata | English | French | English | German |
| Normal Version | Number of Relationships | 253 | 144 | 211 | 177 | 225 | 118 |
| | Number of Entities | 38421 | 40159 | 36508 | 33532 | 38281 | 37069 |
| Dense Version | Number of Relationships | 220 | 135 | 217 | 174 | 207 | 117 |
| | Number of Entities | 68598 | 75465 | 71929 | 66760 | 56983 | 59848 |

**Table 3.** Data sets without inverse relations

| Dataset | Number of entities | Number of relationships | Number of entity pairs in the training set | Number of entities pairs in the validation set | Number of entity pairs in the test set |
|---|---|---|---|---|---|
| WN18RR | 40943 | 11 | 86835 | 3034 | 3134 |
| FB15k37 | 14541 | 237 | 272115 | 17536 | 20466 |

## 5.3. Experimental results and analysis

The metrics commonly used for the evaluation of knowledge graph embedding tasks are MRR, Hit@1, Hit@10, etc. Since the performance of an architecture is objective and does not vary substantially with the measurement metrics, next, we perform the result comparison with Hit@1 as the main evaluation metric for the search process demonstration.

For a fair comparison, we take the same path sampling scheme and data partitioning scheme on each dataset.

The following line graph gives a comparison of the performance of the original algorithm and the improved algorithm searching out the architecture on different datasets. The solid blue line represents the architecture searched by the original algorithm, the dashed green line represents the improved algorithm, and the red implementation represents the optimal neural architecture currently designed by humans on this dataset. As can be seen from the line graph, the performance of the optimal architectures searched in the DBP_WD, EN_DE datasets after adding the clustering algorithm are close to or higher than the performance of the optimal architectures searched by the original search algorithm. Among them, the performance of the optimal architecture searched in the dense version of DBP-WD dataset is better than the performance of the current human hand-designed architecture. The algorithm searched the first thirty search rounds and found architectures that were very close to the performance of the human hand-designed optimal architecture. This demonstrates that our improvement of the search strategy is effective and enables the algorithm to search for better performing architectures based on the original

search architecture.

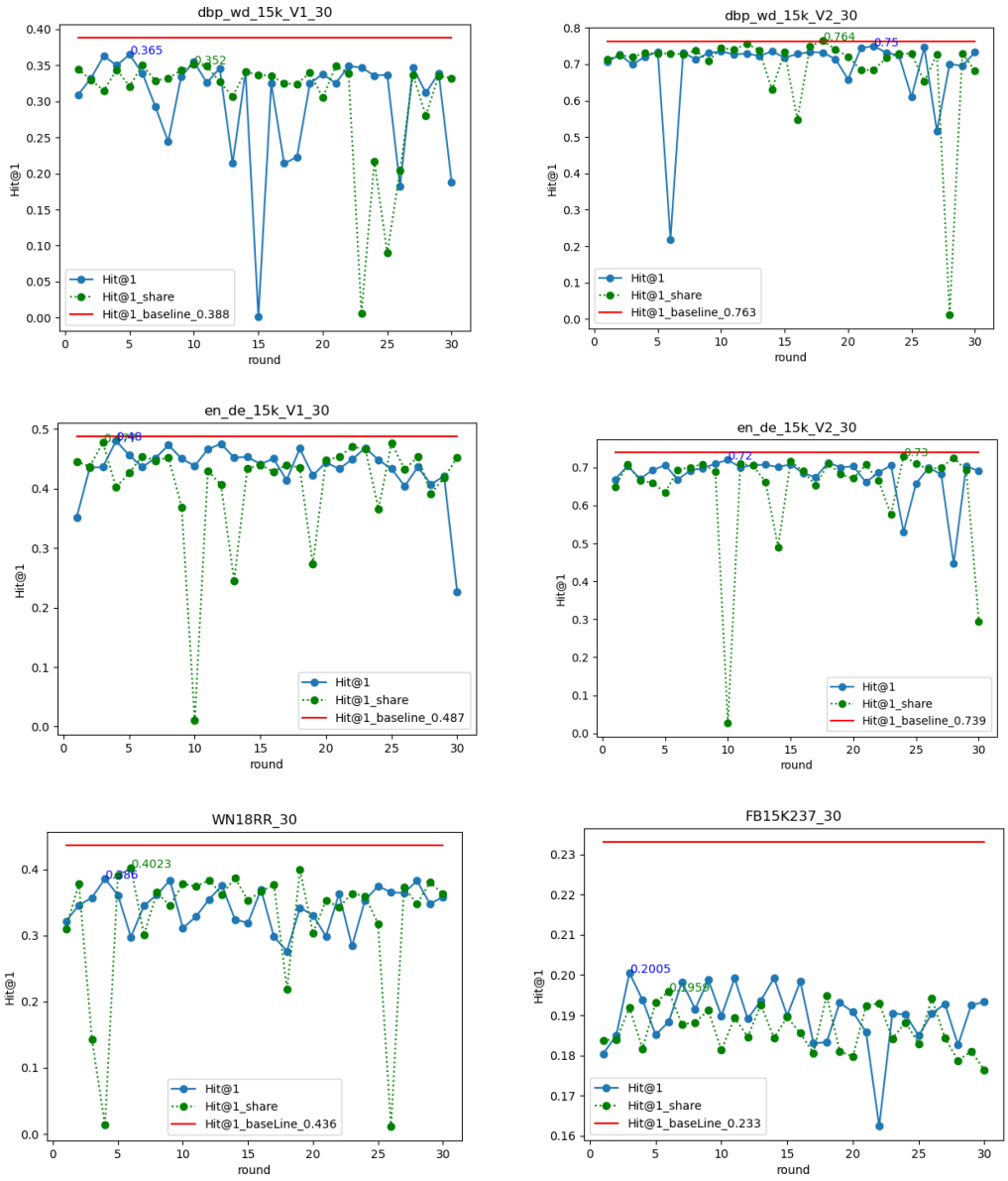


**Figure 2.** Experimental results and demonstration

## 6. Conclusion and outlook

Knowledge graph embedding is a very important form of knowledge graph representation. Existing knowledge graph embedding models have limited scalability and are required for both downstream tasks and datasets of knowledge graphs. Existing approaches combine neural network architecture search with knowledge graph embedding. We further perform algorithm optimization based on the Interstellar algorithm to improve the initialization method of the probability distribution matrix of microscopic parameters. After validation on different downstream tasks and datasets of the knowledge graph, our improvements are found to be effective and improve the search efficiency. In reference [7], researchers applied neural architecture search techniques to find scoring functions for knowledge graph tasks, and in the future, if we can combine the searched neural networks with the searched scoring functions, we believe that we can not only solve the tasks in knowledge graphs better, but also improve the automation to a great extent. Most previous studies on knowledge graph embedding learning have focused on triad-based models, in this paper we emphasize the use of relational paths to learn from knowledge graphs, using neural networks to balance short-term and long-term information in the paths, and if this approach is applied to inference tasks of knowledge graphs it is believed that it will also perform well.

## References


[1] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, et al. Learning Transferable Architectures for Scalable Image Recognition [A], IEEE Conference on Computer Vision and Pattern Recognition[C], Conference on Computer Vision and Pattern Recongnition press, 2018: 8697-8710.

[2] Thomas Elsken, Hendrik Jan, and Frank Hutter. Neural architecture search: A survey[J]. J Mach Learn Res, 2019, 20: 55:1-55:21.

[3] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari,et al. Adaptive stochastic natural gradient method for one-shot neural architecture search. In International Conference on Machine Learning [J], 2019:171-180.

[4] HanXiao Liu, Karen Simonyan, and Yiming Yang. Yao: Differentiable architecture search[R]. In International Conference on Learning Representations ,2019.

[5] Auer S, Christian Bizer, Georgi Kobilarov, et al. DBpedia: A nucleus for a web of open data[A]. ISWC/ASWC [C], 2007:722-735.

[6] Yongqi Zhang, Quanming Yao, Lei Chen. Interstellar: Searching Recurrent Architecture for Knowledge Graph Embedding[A]. H Larochelle. Advances in Neural Information Processing Systems[C], Curran Associates, Inc: 2020: 10030-10040.

[7] Quanming Yao, Ju Xu,Weiwei Tu, et al. Efficient neural architecture search via proximal iterations. American Association for Artificial Intelligence press[C], 2020:6664-6671.

[8] Lingbing Guo, Zequn Sun and Wei Hu. Learning to exploit long-term relational dependencies in Knowledge graphs. International Conference on Machine Learning [J], 2019:2505-2514.