Technical Report: Bayesian Optimization on Detours

Qingfeng Xiang Yiming Lee Zhenyu Lin Yingxia Shao

Beijing University of Posts and Telecommunications, China

1 Introduction

This technical report is a brief introduction to the iDMG team's 2022 GNNetworking Challenge's solution, the fourth place in evaluation phase. We design a approach named Bayesian Optimization on Detours(BOD) to create networks up to 10 nodes for GNN to scale to larger networks. We formulate training data generation as a black-box optimization problem and our solution achieves MSE=10.21327% on the test set through 100 training data with no more than 10 nodes, which strictly meets the constraints.

2 Problem Formulation

We define the task of this competition as an optimization problem:

$$\min_{\vec{x}\in DOM} loss(\vec{x}), \tag{1}$$

where the objective function $loss(\vec{x})$ represents the MSE loss value of the RouteNet-Fermi model on the validation set trained by a set of training data which is generated according to the hyperparameter $\vec{x}$.

The vector $\vec{x}$ contains 17 hyperparameters and gives a fine-grained control over many aspects of training data generation, and its parameter space DOM is manually specified (shown in Table II).

In the next subsections, we describe how the hyperparameter $\vec{x}$ controls training data generation. And we'll introduce the technique we used in solving this optimization problem: Parallel Bayesian Optimization.

3 BOD's Overview

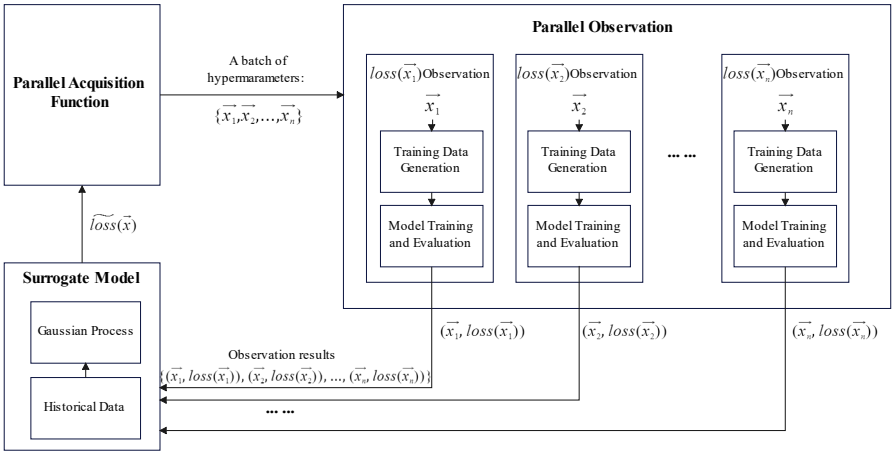

Fig. 1: BOD's Overview

Fig. 1 shows Bayesian Optimization on Detours(BOD)'s overview, BOD follows a Bayesian Optimization framework, including a surrogate model, a parallel acquisition function and an objective function $loss(\vec{x})$ that can be observed in parallel.

During execution, the surrogate model first models $loss(\vec{x})$ as a Gaussian Process according to historical data, and the acquisition function selects a batch of observation points that may reduce objective function's value. Then BOD evaluates these observation points in parallel, and updates the surrogate model according to the observation results. The process above is repeated until the preset maximum number of observations is reached. Finally, BOD takes the lowest $loss(\vec{x})$ and its corresponding $\vec{x}$ as the output.

4 Objective Function

As mentioned above, we abstract the entire process of training data generation, RouteNet-Fermi model

training and evaluation into an objective function in (1). Given $\vec{x}$, the observation of $loss(\vec{x})$ consists of two stages: training data generation and model evaluation.

Training data generation

Our training data construction scheme is improved on the basic data construction scheme provided by the competition, and the main process remains unchanged.

1). Topology generation

At this stage, we generate 10 graph topologies with 10 nodes respectively. Machine learning is based on the assumption of independence and identical distribution. Therefore, the data on the training set and the test set should maintain similar distribution as far as possible to get good performance.

By analyzing the data distribution on the validation set, we find that the degree of graph nodes roughly follows the normal distribution, the link bandwidth is sampled from 4-5 discrete values, and the node's buffer size is always sampled from 4 discrete values randomly. Therefore, we have manually specified three different kinds of degrees for 10 nodes, representing sparse, dense and medium graph, and we specify four discrete bandwidth values for each type of graph. Table I shows the node degrees of these three types of graphs and their corresponding possible values of link bandwidth and buffer size.

Table I: Parameters for Generating Topology

Graph Type	Graph's Quantity	10 Nodes' Degrees	Bandwidth	Buffer Size
Sparse	3	[1,2,2,2,3,3,3,3,3,4]	$[bw_{11}, bw_{12}, bw_{13}, bw_{14}]$	[8k,16k,32k,64k]
Medium	4	[2,3,3,3,3,3,3,3,4,5]	$[bw_{21}, bw_{22}, bw_{23}, bw_{24}]$	[8k,16k,32k,64k]
Dense	3	[2,3,3,4,4,4,4,4,5,5]	$[bw_{31}, bw_{32}, bw_{33}, bw_{34}]$	[8k,16k,32k,64k]

Although we make our best effort to describe the distribution of validation set as accurately as possible, the generalization performance is poor because many parameters are determined manually, which leads to serious overfitting phenomenon. Therefore, we add noise to the generation process of bandwidth and buffer size, which follow two uniform distribution respectively controlled by hyperparameter $bandwidth\_range$ and $buffer\_range$, to gain model's generalization ability.

Apart from link bandwidth and buffer size, we randomly choice scheduling policy and generate buffer weights, but use a hyperparameter $schedulingWeight\_factor$ to determine how different the weights are.

In a brief summary, there are totally 12 hyperparameters for bandwidth(bw_{11} to bw_{34}, mentioned in Table I), 3 hyperparameters for noise($bandwidth\_range$, $buffer\_range$ and $buffer\_step$) and a hyperparameter for buffer weights($schedulingWeight\_factor$).

2). Traffic matrix generation

We analyzed the traffic matrix's distribution of the validation dataset and generated a similar distribution in the training data.

3). Route generation

On the validation set, we observe that the maximum route length between 50 nodes networks and the 300 nodes networks are very similar, which is basically 8 to 10. However, a 10 nodes network is difficult to have such a long route. We try to lengthen the routing path in the 10 nodes network to make it more consistent with the larger network. Among the searched detours with a specified length, we well use genetic algorithm to select one, and try to balance the load of each link, which is similar to the situation of network snapshots in reality.

4). OMNet++ simulation

As required by the rules, we directly use the simulator image provided by the sponsor to generate the final training data.

Model evaluation

We didn't adjust any preset parameters, and directly trained RouteNet-Fermi for 20 epochs. We will select the best one of 20 epochs as the final objective function's loss value.

5 Parallel Bayesian Optimization

Since the objective function is an expensive black-box function to evaluate, we employ Bayesian

Optimization to solve this optimization problem. By modeling $loss(\vec{x})$ as a Gaussian Process, the surrogate model can predict the confidence interval of $loss(\vec{x})$ according to historical data. Then BO can select a batch of next points to observe smartly by using a parallel acquisition function.

Surrogate model

In our method, we model $loss(\vec{x})$ as a Gaussian Process $\widetilde{loss}(\vec{x})$, and describe $\widetilde{loss}(\vec{x})$ with a mean function $\mu(\cdot)$ and covariance kernel function $k(\cdot)$. We let $\mu(\cdot)$ always be zero and use $Matern5/2$ as its kernel function.

Parallel Acquisition function

To maximize the number of observations, we use an asynchronous parallel acquisition function as follows[1]:

$$\alpha_{UCB}(\vec{x}) = \mu(\vec{x}) + \kappa\sigma(\vec{x}), \tag{2}$$

$$\overrightarrow{x_{next}} = \arg\max_{\vec{x}\in DOM}\left\{\alpha_{UCB}(\vec{x})\prod_{i}^{k-1}\phi(\vec{x}|\overrightarrow{x_i})\right\}, \tag{3}$$

where $\alpha_{UCB}(\vec{x})$ is a classical UCB acquisition function, $\mu(\vec{x})$ and $\sigma(\vec{x})$ are the mean and standard deviation of the Gaussian Process, κ is a parameter that controls the trade-off between exploration and exploitation. This parallel acquisition function uses local penaliser $\phi(\vec{x}|\overrightarrow{x_i})$ to select a batch of observation points sequentially and can be asynchronous.

Through asynchronous parallel BO technology, we can observe the objective function more than 150 times per day on an Intel(R) Xeon(R) Gold E5220 CPU, thus finding good hyperparameters

REFFERENCE

[1] Alvi A, Ru B, Calliess J P, et al. Asynchronous Batch Bayesian Optimisation with Improved Local Penalisation[C]//International Conference on Machine Learning. PMLR, 2019: 253-262.

Appendix

A. Hyperparameters

Table II: 17 hyperparameters for training dataset generation

Hyperparameter	Range	Default Value	Description
bandwidth11	[18,30]	20	Descrete bandwidth value for graph 1-3
bandwidth12	[30,40]	30	
bandwidth13	[46,56]	51	
bandwidth14	[84,94]	89	
bandwidth21	[18,25]	20	Descrete bandwidth value for graph 4-7
bandwidth22	[30,40]	35	
bandwidth23	[46,56]	51	
bandwidth24	[84,94]	89	
bandwidth31	[18,25]	20	Descrete bandwidth value for graph 8-10
bandwidth32	[30,40]	30	
bandwidth33	[46,56]	51	
bandwidth34	[84,94]	89	
bandwidth_range	[1,3]	1	Bandwidth noise
buffer_range	[0,2]	0	Buffer size noise
buffer_step	[1,4]	1	Buffer size sampling step
schedulingWeight_fa	[0,75]	25	Buffer weight factor
route_len_plus	[0,6]	3	Route length threshold