Lecture Notes in Electrical Engineering 1207

Xiaoduo Li
Xun Song
Yingjiang Zhou *Editors*

Proceedings of 2023 7th Chinese Conference on Swarm Intelligence and Cooperative Control

Swarm Decision and Planning Technologies

Springer

Lecture Notes in Electrical Engineering 1207

Xiaoduo Li · Xun Song · Yingjiang Zhou
Editors

Proceedings of 2023 7th Chinese Conference on Swarm Intelligence and Cooperative Control

Swarm Decision and Planning Technologies

Springer

Editors
Xiaoduo Li
Sino-French Engineer School
Beihang University
Beijing, China

Xun Song
Sun Yat-sen University
Beijing, China

Yingjiang Zhou
College of Automation and College of Artificial Intelligence
Nanjing University of Posts and Telecommunications
Nanjing, Jiangsu, China

ISSN 1876-1100 ISSN 1876-1119 (electronic)
Lecture Notes in Electrical Engineering
ISBN 978-981-97-3335-4 ISBN 978-981-97-3336-1 (eBook)
https://doi.org/10.1007/978-981-97-3336-1

Contents

DeepTQ: Predictive TSN Switch Queue Length Based on Deep Learning

Yanxin Jia[1,3], Long Xu[1,2,3](✉), Wei Xiong[1,2,3], Xin Wang[4], Zhijun Shang[4], and Zijun Yuan[1]

[1] 3onedata Co., Ltd., Shenzhen, China
long.xu.rmit@gmail.com
[2] Ministry of Education, School of Automation and the Key Laboratory of Measurement and Control of Complex Systems of Engineering, Southeast University, Nanjing, China
[3] 3onedata Qitong Co., Ltd., Shanghai, China
[4] University of Chinese Academy of Sciences, Beijing, China

Abstract. With the rapid development of network technology, TSN (Time-Sensitive Networking) is in the stage of rapid development. It needs to ensure the deterministic transmission of time-sensitive flow, further improve the network throughput, and meet the increasing traffic demand. SDN (Software-Defined Networking) can realize the automatic management of network resources by decoupling the control plane and the data plane. Through global network information, it uses efficient route scheduling to achieve load balancing and improve network throughput. However, the prerequisite for route scheduling is that accurately predict the queue length of TSN switch, so that route scheduling can be carried out according to the predicted results. To address the queue length prediction problem of TSN switch, DeepTQ (Predictive TSN Switch Queue Length Based on Deep Learning) is proposed, which predicts the queue length of TSN switches in the next time slot using deep learning methods. It consists of the feature selection phase and the model prediction phase. In DeepTQ, the feature selection phase uses the Spearman rank correlation coefficient to remove redundant features in the datasets. The model prediction phase uses an improved Transformer Encoder and GRU (Gate Recurrent Unit) to construct the prediction model. Simulation results demonstrate that DeepTQ achieves superior performance compared to existing methods and individual components contribute to its overall effectiveness.

Keywords: Time-Sensitive Networking · Gate Recurrent Unit · Transformer · Software-Defined Networking

1 Introduction

As Operation Technology (OT) and Information Technology (IT) continue to advance, network applications require higher-quality traffic transmissions for tasks such as industrial control, telemedicine surgery and etc. [1]. These applications demand end-to-end latency below 10 ms and jitter of the microsecond level. However, issues such as complex fieldbus, different real-time requirements, and transmission mechanisms have hindered IT-OT interaction [2]. To address these problems, the IEEE 802.1 group developed Time-Sensitive Networking (TSN), a set of link-layer communication standards that enable real-time, deterministic, and compatible transmission of Ethernet, industrial control data, and Ethernet data.

Multiple applications connected to the network will cause a sharp increase in the data flow, and put more pressure on the network infrastructure. Considering the sudden flow of packets, traditional routing algorithms cannot effectively avoid network congestion. Developers have developed the concept of "Software-Defined Networking" (SDN) to address the need for flexible networks. SDN can realize the automatic management of network resources by decoupling the control plane and the data plane. Through global network information, it uses efficient route scheduling to achieve load balancing and improve network throughput. Wang *et al.* [3] based on time-sensitive software-defined network (TSSDN) architecture, use dimension reduction, feature selection, and LSTM (Long Short Term Memory) prediction model to predict the queue length of TSN switch. Experiments show that this method can obtain better experimental results. Yao *et al.* [4] used machine learning methods to assist route load balancing. They used dimensionality reduction, queue utilization prediction and route load balancing to achieve route load balancing. The simulation results show that queue utilization prediction is better than the traditional Bellman-Ford routing strategy. In the above studies, the queue length of the switch is predicted, and then the length is provided for the subsequent route scheduling.

The prediction of switch queue length depends on histroy information. To improve the performance of the prediction model, this paper introduces Transformer and GRU two time series methods for learning. It is DeepTQ (Predictive TSN Switch Queue Length Based on Deep Learning), which for predicting the next time slot queue length of TSN switches based on SDN architecture. It collects link information (such as queue length, throughtput etc.) of the entire network from data plane and the SDN controller can make better routing decisions and avoid network congestion, which leads to improved network performance and reduced latency. The main contributions of this paper are summarized as follows:

1) This paper proposed a method DeepTQ to predict the queue length of TSN switches in the next time slot based on SDN architecture. It combines an improved Transformer Encoder and GRU, which is the first time such a combination has been used for queue length prediction.
2) DeepTQ consists of two phases: the feature selection phase and the model prediction phase. The feature selection phase filters out redundant features from the link information collected in the data plane, while the model prediction phase predicts the next time slot queue length of TSN switches.
3) The experiments conducted on datasets obtained by OMNET++ simulation demonstrate that DeepTQ outperforms other baseline methods and achieves

the best performance. DeepTQ can improve the maximum performance measure of MSE, MAE, and RMSE by up to 1.45%, 3.11%, and 3.70%, respectively. Furthermore, the feature selection phase and the GRU in the model prediction phase can also contribute to the overall performance improvement of DeepTQ.

The paper is organized as follows. Section 2 introduces the background of TSN and SDN. Section 3 introduces the proposed method DeepTQ, which consists of two phases: the feature selection phase and the model prediction phase. Section 4 introduces the implementation and evaluation, which specifically includes the collection of datasets, the parameter setting of the model, the performance measure of the prediction model, and the experimental results and ablation study of the DeepTQ. Section 5 concludes the paper and describes future work.

2 Background

2.1 TSN

In the era of Industry 4.ch10, the information based on OT needs to interact with the application software of IT [5], but there are many problems existing, such as complex fieldbus, different real-time requirements, and different network transmission mechanisms. To address these issues, the IEEE 802.1 working group lead the development of a set of link-layer communication standards, namely TSN. It can meet the real-time and determinism of industrial control and compatibility with Ethernet to realize the mixed transmission of industrial control data and Ethernet data [6].

To maintain compatibility with standard Ethernet, the TSN is built based on IEEE 802.1Q Virtual Local Area Network (VLAN). The characteristics are defined by inserting 4 bits in the standard Ethernet frame [7].

1) Tag Protocol Identifier is 0X810. It identifies the Ethernet supports 802.1Q tag.
2) Priority Code Point defines 8 priority levels. Priority 0 is the lowest and used for traditional best-effort traffic, priority 7 is the highest and used for critical routing or network management functions [8].
3) Drop Eligible Indicator identifies the frame is discardable.
4) VLAN Identifier is the identification number of the VLAN network.

TSN will distinguish the importance of network traffic by priority and perform traffic scheduling and network configuration according to different application scenarios.

2.2 SDN

Software-Defined Networking (SDN) is a programmable network architecture that separates the control plane from the data plane. It consists of three parts:

data plane, control plane, and application plane [9]. Among them, the bottom layer is the data plane, which mainly composes the network device and data links forming the basic forwarding network. The middle layer is the control plane, which is the control center of the SDN system, mainly responsible for the generation and control of routes within the network and at the network boundary, maintaining the network change state, etc. The top layer is the application plane, which is mainly responsible for deploying various web applications developed by users according to their needs.

In SDN controller, it implements the underlying hardware control and on-demand provisioning of network resources through a programmable software platform. Whenever a data flow comes to the switch, the routing algorithm on the control plane starts planning the route and then generates a flow table that is sent down to the switch to complete packet forwarding. According to the ideas of Wang *et al.* [3], in this paper, the controller collects the switch's real-time status from the data plane and implements switch queue length prediction for the next time slot. The control plane plans routes based on the prediction results and selects the next-hop address to avoid network congestion, then generates flow tables and sends them down to the switch to complete packet forwarding.

3 DeepTQ

In this paper, the proposed method DeepTQ consists of two phases: the feature selection phase and the model prediction phase. The feature selection phase is used to filter the network link information collected in the data plane, removing redundant features and improving the performance of the prediction model. The model prediction phase involves constructing the prediction model based on the improved Transformer Encoder and GRU, which is used to achieve the prediction of the next time slot queue length of TSN switches.

3.1 Feature Selection Phase

DeepTQ employs the Spearman rank correlation coefficient as the feature selection method, which is utilized to measure the correlation between two variables. This method is advantageous because it does not impose any requirements on the distribution of the original variables and is applicable to variables that do not follow bivariate normal distribution, have unknown distributions, or lack rank information [10]. It has been widely used in various disciplines [11] [12].

Suppose there are two variables X, Y, the number of them is n, and the i-th values are X_i, Y_i. Sort X, Y in ascending or descending order to obtain two new sets x, y, respectively, where x_i is the ranking of X_i in X and y_i is the ranking of Y_i in Y. A ranking difference set d will be obtained by subtracting the corresponding elements in the combined x, y, where $d = x - y$, $1 \leq i \leq n$, then the Spearman rank correlation coefficient between the random variables x, y is defined as [13] [14]:

$$\rho = 1 - \frac{6\sum_i^n d_i^2}{n\left(n^2-1\right)} \tag{1}$$

The link information from TSN switches is utilized for feature selection. Among these, the collected link information includes send, arrival, size, queuetime, node, priority, interval, traffic, throughput, and y. A detailed description is provided in Sect. 4.1 Table 2. Figure 1 depicts the Spearman correlation coefficient plot among these features. In the lower triangle, the correlation values are displayed, while the upper triangle visualizes the ovals. The flatter the ellipse, the darker the color, indicating that the absolute value of the correlation coefficient is closer to 1. Following Jiarpakdee *et al.*'s recommendation [15] , the feature selection phase eliminates features with an absolute value greater than or equal to 0.7. Ultimately, the features used in the model prediction phase are send, priority, interval, traffic, and y

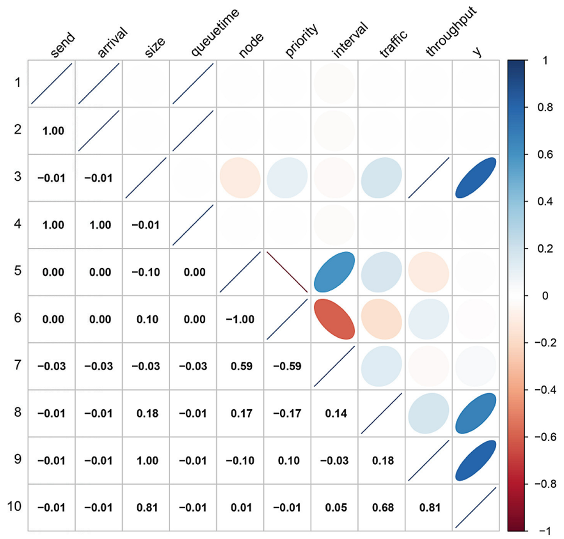

Fig. 1. Spearman correlation coefficient plot

DeepTQ relies on data from the previous N moments when predicting the switch queue length for the next time slot. Therefore, the input feature is represented as $[M_f, N_t]$, where M_f represents the number of features, and N_t represents the factor_time denoting the features from the previous N moments. The specific configuration is detailed in Sect. 4.2.

3.2 Model Prediction Phase

After the aforementioned feature selection phase, the variables to be used for modeling have been determined. These selected variables are utilized by the model constructed during the model prediction phase to predict the queue length for the next time slot. DeepTQ implements model prediction based on an improved Transformer Encoder with GRU. The specific structure is depicted in Fig. 2. It is composed of N Encoders, and the structure of each layer of an encoder is as follows.

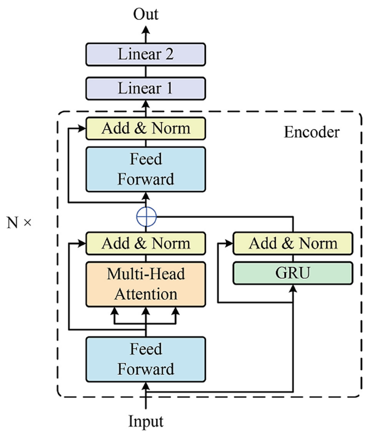

Fig. 2. Structure of the prediction model

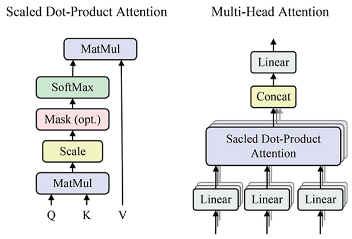

Fig. 3. Multi-Head Attention [16]

1) Feed Forward. The Feed Forward layer consists of two fully-connected layers, where the hidden size is 1024, the activation function is $ReLU$, and the output size is 128, the activation function is *sigmoid*. It maps the data to the high-dimensional space and then to the low-dimensional space, which can learn more represent the relationship between each feature. DeepTQ incorporates the Feed Forward layer after the input to enhance the performance of subsequent attention mechanism learning.

2) Multi-head Attention.
Attention is a weighted summation mechanism that computes value based on query. DeepTQ uses Multi-Head Attention in Transformer Encoder [16]. Where Attention is also known as scaled dot product attention, as shown in the left of Fig. 3. The input Q, the dimension of K is d_k, and the dimension of V is d_v. Firstly, the dot product of q and all keys is divided by the root number d_k, and then the $softmax$ function is applied to get the weights of all v. The calculation formula is

$$Attention\left(Q,K,V\right) = soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2}$$

Vaswani *et al.* found that multi-head attention enables the model to learn information from different representation subspace at different locations, and results in better model performance [16]. So, DeepTQ uses multi-head attention to merge h attentions with the structure shown on the right side of Fig. 3, and the computational procedure is

$$Multi - Head\,(Q, K, V) \,=\, Concat\,(head_1, ..., head_h)\, W^Q, \tag{3}$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right). \tag{4}$$

DeepTQ uses different number of heads for different queues. See the Sect. 4.2.

3) Add & Norm.
Add & Norm use for residual connection followed by batch normalization. Among them, the residual connection [17] [18] is to solve the problem of network degradation and avoid the degradation of model performance due to the increase of network depth. Batch normalization [19] is to do normalization on the same dimensional features of batch samples, which can greatly improve the model training speed and improve the network generalization performance.

4) GRU.
Gate Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that comprises two essential components: an update gate and a reset gate. The update gate regulates the amount of state information that can be preserved from the previous time step. A larger value allows for more retention of previous state information. On the other hand, the reset gate determines the degree to which the current state is blended with the previous information. A smaller value results in more information being disregarded [20].
In DeepTQ, the GRU consists of three layers. The first layer is a GRU with a model dimension of 128 and *ReLU* activation function. The second layer is a dropout layer with a dropout probability of 0.5. The third layer is the output layer, which is a fully connected layer with one unit and a *sigmoid* activation function.

5) Linear.
DeepTQ employs two linear layers to implement the learning of the model. Linear 1 has 128 dimensions and utilizes the *ReLU* activation function. Linear 2 has 1 dimension and utilizes the *sigmoid* activation function, which is used for the final prediction.

DeepTQ consists of N Encoders. The Encoder construct of Feed Forward, Multi-Head Attention, Add & Norm, and GRU. Specifically, the input pass through Feed Forward, Multi-Head Attention, and Add & Norm as the Attention learned information, the input pass through GRU, Add & Norm as the GRU learned information. Then the two pieces of information are summed, through Feed Forward, Add & Norm as the output of an Encoder. According to different training data, the number of Encoders can be adjusted by itself, and finally, the Encoder output is passed into Linear to achieve the final prediction.

When DeepTQ is trained, its parameters are stored locally for the prediction of TSN switch queue length in the controller of SDN. The SDN control plane plans routes based on the prediction results and selects the next-hop address to avoid network congestion, then generates flow tables and sends them down to the switch to complete packet forwarding. In this way, the controller can perform routing based on the real-time queue length, predict the switch queue length to achieve load balancing globally, and avoid network congestion and low bandwidth utilization.

4 Implementation and Evaluation

4.1 Datasets

This paper constructs the TSN simulation environment using OMNET++ [21]. It contains 5 hosts and 1 switch, inspired by the experiments of Wang *et al.* [3], as shown in Fig. 4, where node 1 to node 4 randomly sends packets to node 5. They send three types of flow (TT, AVB and BE) and other detail descriptions as Table 1. To simulate the high-priority bursts data flow as much as possible, the paper uses an exponential function to send the high-priority bursts data flow at 2000us. The remaining priority of flow are simulated using normal and uniform functions.

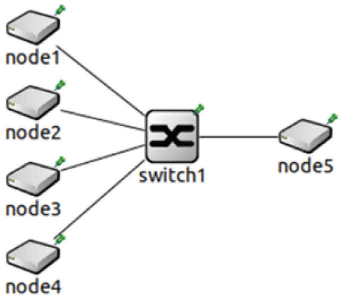

Fig. 4. Experimental simulation topology

Table 1. Description of datasets statistics

Node	Send Interval	Priority	Queue	Data Size
1	exponential(2000us)	7	1	17543
2	exponential(2000us)	6	2	1345
3	normal(2500us,1000us)	5	3	2491
4	uniform(1.25e-4 s,1.25E-3 s)	0	4	7175

Based on the simulation environment, the feature of collected datasets include send, arrival, size, queuetime, node, priority, interval, traffic, throughput and y. These features are explained as shown in the Table 2.

Table 2. Detailed description of each feature

Feature	Description
send	Time when a packet sends the sending node
arrival	Time when a packet reaches the receiving node
size	Size of the sent packet
queuetime	Time when a packet reaches the switch
node	Number of the sending node
priority	Priority of packets sent by the node
interval	Difference between send and arrival
traffic	Number of packets arriving at the switch in the previous time
throughput	Number of successful packet transfers per unit time
y	Number of packets in switch queue at moment

4.2 Setting

1) Datasets Split.
The datasets of different queues were split into training and test sets using an 8:2 ratio. Since the change of queue length is in chronological order, the first 80% data of the datasets uses as the training set and the last 20% data uses as the test set according to the node send time. With this split, historical data are used to predict new data, which is consistent with real application scenarios.
2) Operating System Environment.
The proposed method runs on Windows 10 Operation System (CPU Intel(R) Core(TM) I5-6300HQ and Memory is 24 GB).
3) Model Parameters.
The settings of the prediction model in the model prediction phase are as follows.
   a. Optimizer.
   The model uses Adaptive Moment Estimation (Adam) optimization algorithm to optimize the model parameters. Optimizer parameter settings are consistent with Transformer and specific is $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$ [16].
   b. Dropout.
   The model apply dropout before data is added to the sublayer input and normalized. The rate is 0.3.
   c. Loss Function.
   The model use MSE, the most commonly used function in regression problems [3] [22], as the loss function.
   d. Parameters.
   The parameters of Deep TQ include the number of attention (num_heads), the number of Encoder layers (num_layers), the number of training iteration for all batches (epochs) and the features of the previous N moments (factor_time). According to the preliminary experimental exploration, the

model set num_heads=2, num_layers=2 and epoch=3 in queue 1, queue 2 and queue 4. Due to the different data size, factor_time of queue 1 and queue 2 is 25 and factor_time of queue 4 is 5. Set the parameters of queue 3 to num_heads=8, num_layers=6, epochs=10, and factor_time=10.

4.3 Measure

DeepTQ uses to predict the queue length of TSN switches, which is a regression problem. Therefore, this paper choose three performance measures commonly used in regression problems to evaluate the performance of the method [3] [22]. In following formula, y_i represents the real value, $\hat{y_i}$ represents the predicted value and m represents the size of samples.

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y_i})^2. \tag{5}$$

$$MAE = \frac{1}{m}\sum_{i=1}^{m}\mid (y_i - \widehat{y_i}) \mid. \tag{6}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y_i})^2}. \tag{7}$$

The value range of MSE (Mean Square Error), MAE (Mean Absolute Error), and RMSE (Root Mean Square Error) is from 0 to infinity. The smaller the value, the smaller the error between the real value and the predicted value, and the better performance of the model.

4.4 Results

In previous research efforts, only the experiments conducted by Wang *et al.* [3] were similar to the ones presented in this paper. They constructed a topology for sending three types of data flow, and then used LSTM to predict switch queue length. Based on this, this paper builds the same topology and uses LSTM as a baseline. In particular, they used traffic type and queue length as input to achieve the prediction of queue length in the next time slot.

To ensure the fairness of the experiment, the input of the factor_time by LSTM was kept consistent with that of DeepTQ, as detailed in Sect. 4.2. The results of DeepTQ and the baseline LSTM are presented in Table 3, with the best results highlighted in bold. As shown in the table, DeepTQ exhibits only a slight improvement on queue 1, with an increase of only 0.01% in performance. However, it shows significant improvements on the other three queues, with MSE improving up to 1.45%, MAE up to 3.11%, and RMSE up to 3.70%.

Table 3. The results of DeepTQ and baseline

Queue	Method	MSE (%)	MAE(%)	RMSE(%)
1	LSTM	4.01	15.7	20.02
	DeepTQ	3.33	14.45	18.25
	Improvement	**0.68**	**1.25**	**1.77**
2	LSTM	4.58	16.9	21.39
	DeepTQ	3.13	13.79	17.69
	Improvement	**1.45**	**3.11**	**3.70**
3	LSTM	6.17	19.4	24.84
	DeepTQ	4.85	17.52	22.03
	Improvement	**1.32**	**1.88**	**2.81**
4	LSTM	2.09	11.04	14.47
	DeepTQ	2.08	11.18	14.43
	Improvement	**0.01**	-0.15	**0.04**

4.5 Ablation Study

To verify the effectiveness of different components in DeepTQ, this paper conducts ablation experiments on datasets. Table 4 shows measures of MSE, MAE and RMSE. The table provides a comparison of the performance measures of DeepTQ, DeepTQ without the feature selection phase and DeepTQ without the GRU in the model prediction phase. DeepTQ-NF represents DeepTQ without the feature selection phase, while TransE denotes DeepTQ without the GRU and its associated Add & Norm layer in the model prediction phase. The best performance is indicated in bold.

Table 4. The result of ablation study

Queue	Method	MSE (%)	MAE(%)	RMSE(%)
1	DeepTQ-NF	3.57	14.54	18.89
	TransE	**3.26**	14.64	**18.05**
	DeepTQ	3.33	**14.45**	18.25
2	DeepTQ-NF	5.69	17.52	23.86
	TransE	5.02	16.47	22.41
	DeepTQ	**3.13**	**13.79**	**17.69**
3	DeepTQ-NF	5.60	17.92	23.67
	TransE	9.07	26.23	29.78
	DeepTQ	**4.85**	**17.52**	**22.03**
4	DeepTQ-NF	2.29	12.10	15.12
	TransE	2.08	**11.08**	**14.42**
	DeepTQ	**2.08**	11.18	14.43

Based on the results in the table, it can be concluded that:

1) The feature selection phase of DeepTQ is crucial in improving performance measures across all queues. Notably, the most significant improvement can be observed in queue 2, as indicated in Table 4. The MSE, MAE, and RMSE metrics show increases of 2.56%, 3.73%, and 6.17%, respectively.
2) The inclusion of GRU in the model prediction phase also contributes to the overall enhancement of model performance. While there are slight performance variations in queues 1 and 4, these differences remain below 0.2%. Considering the substantial improvements observed in other queues and the relatively lower performance measures in queues 1 and 4, it can be concluded that the incorporation of GRU positively impacts the overall performance of DeepTQ.

5 Conclusion and Future Work

Through global network information, SDN can achieve load balancing and improve network throughput using efficient route scheduling. However, the prerequisite for route scheduling is that accurately predict the queue length of TSN switch. This paper proposed a method DeepTQ, which consists of two phases: the feature selection phase and the model prediction phase. The feature selection phase filters the network link information collected in the data plane to remove redundant features and improve the performance of the prediction model. The model prediction phase uses an improved Transformer Encoder and GRU to construct the prediction model for the queue length of TSN switches in the next time slot.

To evaluate the effectiveness of the proposed method, the paper simulates a real network environment through OMNeT++ and uses three performance measures (MSE, MAE, and RMSE) to evaluate the prediction model. The results show that DeepTQ outperforms other methods, and the feature selection phase and the GRU of the mode prediction phase can further improve the performance. Specifically, DeepTQ achieves a model performance of 3.33%, 3.13%, 4.85%, and 2.08% for each queue in terms of MSE, 14.45%, 13.79%, 17.52%, and 11.18% for each queue in terms of MAE, and 18.25%, 17.69%, 22.03%, and 14.43% for each queue in terms of RMSE.

In future work, we plan to develop a general prediction model that can adapt to different topologies and traffic flows and can be easily migrated to the corresponding SDN controller.

Acknowledgment. The research was supported by the Shanghai Pujiang Program (Grant No. 21PJ1422000) and National Key R&D Program of China (Grant No. 2022YFB290 1100).

References

1. Wang, S., Yin, S., Lu, H., Zhang, J.: Survey of control and management mechanisms for time-sensitive network. J. Netw. Inf. Secur. **7**(6), 11 (2021)
2. Bello, L.-L., Steiner, W.: A perspective on IEEE time-sensitive networking for industrial communication and automation systems. In: Proceedings of the IEEE, vol. 107(6), pp. 1094–1120 (2019)
3. Wang, X., Shang, Z., Xia, C., Cui, S., Shao, S.: TSN switch queue length prediction based on an improved LSTM network. In: Wireless Communications and Mobile Computing, pp. 1–13 (2021)
4. Yao, H., Yuan, X., Zhang, P., Wang, J., Jiang, C., Guizani, M.: Machine learning aided load balance routing scheme considering queue utilization. In: IEEE Transactions on Vehicular Technology, vol. 68(8), pp. 7987–7999 (2019)
5. Deng, L., Xie, G., Liu, H., Han, Y., Li, R., Li, K.: A survey of real-time ethernet modeling and design methodologies: from AVB to TSN. In: ACM Computing Surveys (CSUR), vol. 55(2), pp. 1–36 (2022)
6. Li, Z., Yang, S., Yu, J., Deng, Y., Wan, H.: Survey of deterministic transmission techniques in time-sensitive networks. J. Softw. **33**(11), 4334–4335 (2022)
7. Song, H.-Z.: A survey of time-sensitive network technologies. In: Automatic Measuring Instrument, vol. 41(2), pp. 1–9 (2020)
8. IEEE standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: frame preemption. IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014) (2016)
9. Fei, H., Qi, H., Ke, B.: A survey on software-defined network and OpenFlow: from concept to implementation. In: IEEE Communications Surveys & Tutorials, vol. 16(4), pp. 2181–2206 (2014)
10. Bishara, A.-J., Hittner, J.-B.: Testing the significance of a correlation with non-normal data: comparison of pearson, spearman, transformation, and resampling approaches. In: Psychological methods, vol. 17(3), pp. 399 (2012)
11. Liu, D., Cho, S.-Y., Sun, D.-M., Qiu, Z.-D.: A Spearman correlation coefficient ranking for matching-score fusion on speaker recognition. In: TENCON 2010-2010 IEEE Region 10 Conference, pp. 756–41. IEEE (2010)
12. Xiao, C., Ye, J., Esteves, R.-M., Rong, C.: Using Spearman's correlation coefficients for exploratory data analysis on big dataset, vol. 28(14), pp. 3866–3878 (2016)
13. Spearman, C.: 'footrule' for measuring correlation. British J. Psychol. vol. 2(1), pp. 89 (1906)
14. Spearman, C.: The proof and measurement of association between two things. Int. J. Epidemiol. **39**(5), 1137–1150 (2010)
15. Jiarpakdee, J., Tantithamthavorn, C., Treude, C.: Autospearman: automatically mitigating correlated metrics for interpreting defect models. In: arXiv preprint arXiv:1806.09791 (2018)
16. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
18. Orhan, A.-E., Pitkow, X.: Skip connections eliminate singularities. In: arXiv preprint arXiv:1701.09175 (2017)
19. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456. PMLR (2015)

20. Cho, K., Van M.-B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. In: arXiv preprint arXiv:1409.1259 (2014)
21. Falk, J., et al.: NeSTiNg: simulating IEEE time-sensitive networking (TSN) in OmNet++. In: 2019 International Conference on Networked Systems (NetSys), pp. 1–8. IEEE (2019)
22. Chicco, D., Warrens, M.-J., Jurman, G.: The coefficient of determination r-squared is more informative than SMAPE, MAE, MAPE, MSE, and RMSE in regression analysis evaluation. PeerJ Comput. Sci. **7**(e623) (2021)

Multiplayer Pursuit-Evasion Differential Games with Uncertain Perceptions: A Cumulative Prospect Theory Approach

Hao Yang(✉), Shi Lu, and Bin Jiang

College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
haoyang@nuaa.edu.cn

Abstract. This paper establishes, for the first time, a Cumulative Prospect Theory (CPT)-based multiplayer pursuit-evasion (MPE) differential game framework to capture the subjective irrationality of pursuers and evaders in perceiving adversaries' policies. In the case of deterministic perceptions, a Nash policy can be designed through the MPE differential game formulation. When policy iterations follow a probability distribution, CPT is incorporated into the conventional MPE differential game to model human prospects under risk preference and probability sensitivity. Subsequently, a perceived Nash policy can be designed, aiming to maximize the prospect function. The effectiveness of the proposed results is demonstrated by a numerical example.

Keywords: MPE differential game · uncertain perceptions · CPT · Nash policy

1 Introduction

Differential game theory has emerged as a valuable tool for analyzing strategic interactions in dynamic systems involving multiple players. Within this framework, the study of MPE differential game has garnered significant attention due to its potential applications in various fields [1,2].

The key problem of MPE differential game lies in the formulation of control strategies and decision-making processes for pursuers and evaders, aiming to achieve conflicting objectives of pursuit and evasion [1]. The analysis and resolution of MPE differential game involve the utilization of diverse methodologies including optimal control theory, dynamic programming, and Nash equilibrium concepts [2]. However, it is noteworthy that in many existing studies, the policy design process predominantly concentrates on deterministic updates [1,2]. Nevertheless, the presence of various factors, such as motion constraints, sensing capabilities, and environmental knowledge, may lead to deviations from the optimal policies and prompt pursuers and evaders to select the non-optimal policies [3,4]. Using probability function captures the likelihood of choosing each policy is a conventional method [5].

In MPE differential games, perception is a critical tool to design policies of pursuers and evaders. Especially, this method offers a viable solution to the challenge of players not being able to directly observe adversaries' policies. An advanced perceptual model is "k-level thinking model" leveraged in the work of [6], where players design their current policies by perceiving their opponent's next strategy. Under deterministic perceptions, pursuers and evaders can make the optimal policy based on the actual performance function [7]. However, for the probabilistic decision, it is essential to consider subjective irrationality among pursuers and evaders during maximizing the expected performance. Elements such as risk preference and probability sensitivity play a crucial role in this context [8]. Actually, pursuers prefer to capture evaders with lower performance costs, while simultaneously underestimate large probabilities associated with these capturing policies. On the contrary, pursuers detest the capture with higher performance costs, coupled with an overestimation of the small probabilities associated with such policies. Evaders have an inverse pattern of preferences. With risk preference and probability sensitivity, players' prospects can be evaluated by using CPT rather than relying solely on the conventional expected operator [9]. Although several studies have applied CPT to engineering problems to highlight situations where actual results deviate from theoretical expectations, to the best of my knowledge, *there is currently a lack of research exploring subjective irrationality in MPE differential games.*

Drawing from the aforementioned observations, *this paper, for the first time, incorporates CPT into the framework of MPE differential game, with the aim of capturing subjective irrationality in the context of probabilistic perceptions.* By considering both the optimal and worst policies under deterministic perceptions, a prospect function is formulated to explore the optimal perceived policy under uncertain perceptions. The main contributions are summarized as:

1) In the presence of probabilistic perceptions, a CPT-based MPE differential game is constructed to capture the risk preference and probability sensitivity of pursuers and evaders, in which a specialized expectation operator is applied to evaluate the expected prospect value.
2) The difficulty of optimizing the complex prospect function is addressed by reformulating it as the search for a policy that maximizes or minimizes the weighting sum of the optimal and worst performances, and eventually, the optimal perceived policy is designed by solving coupled Riccati equations.

Notations: $\mathbf{R}$ represents the set of real values; $\mathbf{R}^n$ represents the set of n-dimensional real vectors; $\mathbf{R}^{n\times m}$ represents the set of $n \times m$-dimensional real matrices; $|N|$ represents the number of elements in the set N; $\dot{z}$ represents the time derivation of z; $\nabla V(z)$ and $\frac{\partial V(z)}{\partial z}$ represent the partial derivation of the function $V(z)$ with respect to z; z^T represents the transposition of z.

2 Preliminaries

In this section, the main problems will be formulated after describing a perception-based MPE differential game model, probabilistic decision model.

2.1 MPE Differential Games

Consider MPE differential game with N_1 pursuers and N_2 evaders. The dynamic of each pursuer and evader satisfies

$$\begin{cases} \dot{z}_{i,x} = v_{i,x}, \\ \dot{z}_{i,y} = v_{i,y}, \end{cases} \tag{1}$$

where $z_{i,x}, z_{i,y} \in \mathbf{R}$ represent the position of player i in the coordinate plane $\mathbf{R}^2$, and $v_{i,x}, v_{i,y} \in \mathbf{R}$ represent the speed along the x-axis and y-axis, respectively. Let $z_i := \{z_{i,x}, z_{i,y}\}$ and $v_i := \{v_{i,x}, v_{i,y}\}$. Then, the augmented system for all players can be rewritten as

$$\dot{z} = Bx, z(0) = z_0. \tag{2}$$

where $z := \{z_1, \cdots, z_N\} \in \mathbf{R}^N$ is the system state, $x := \{v_1, \cdots, v_N\} \in \mathbf{R}^N$ is the control policy of pursuers and evaders, where N is the number of all players. The matrix $B = I$ is the input matrix with the appropriate dimension. Obviously, the system (2) is controllable. In order to distinguish pursuers and evaders, the system (2) is rewritten as

$$\dot{z} = B_1 u + B_2 v, z(0) = z_0. \tag{3}$$

$u := \{v_1, \cdots, v_{N_1}\} \in \mathbf{R}^{N_1}$ and $v := \{v_{N_1+1}, \cdots, v_N\} \in \mathbf{R}^{N_2}$ are control policies of pursuers and evaders, respectively. The matrices B_1 and B_2 are input matrices with appropriate dimensions. The goal of pursuers is to minimize the distance between themselves and evaders, meanwhile, to keep as close as possible with their neighboring pursuers in order to achieve team-cooperation, while evaders desire to maximize the distance. Therefore, the performance index is defined by

$$\begin{aligned} J(z,u,v) &= \int_0^{t_f} \Big[\sum_{i\in N_1}\sum_{j\in\mathcal{N}_i}(z_i - z_j)^T P_{ij}(z_i - z_j) + u^T R u - v^T S v\Big] dt \\ &= \int_0^{t_f} [z^T P z + u^T R u - v^T S v] dt, \end{aligned} \tag{4}$$

where $t_f > 0$ is the terminal time, matrices $P \geq 0$, $R > 0$ and $S > 0$. Due to the coupled nature of the performance index and the system dynamics, there is a plethora of different approaches to the "solution" of a differential game. The most common one, which constitutes an equilibrium concept, i.e., the pure-strategy Nash equilibrium (NE) solution.

Definition 1. ***(NE)*** *A tuple of input policies (u^*, v^*) constitutes a Nash equilibrium of MPE differential game if it satisfies $J(u^*, v) \leq J(u^*, v^*) \leq J(u, v^*)$ for any policies u and v.* □

To ensure the designed policy within a desired performance range, an underlying assumption is introduced as follows. This assumption serves two primary purposes: firstly, it ensures that the designed policy adheres to the predefined performance criteria, and secondly, it mitigates the issue of gain matrix parameter explosion during designing the worst policies for pursuers and evaders, as discussed in the subsequent sections.

Assumption 1 *The performance $J(u,v) \in [a,b]$, where $a,b \in \mathbf{R}^+$ with $a \leq J(u^*,v^*) \leq b$.* □

Based on "k-level thinking model" [6], one-step perception model is introduced to show the progressiveness of the decision process, wherein players formulate their current policies by perceiving next policies of adversaries, essentially employing a "think of what you think" strategy. This basic one-step model can also be extended to multi-steps perception further. The subsequent mathematical formulation of one-step perception is presented as follows.

$$\begin{cases} N_1 : \min\limits_{u(k+1)} J(u(k), v(k+1)|v(k)), \\ N_2 : \max\limits_{v(k+1)} J(u(k+1)|u(k), v(k)), \end{cases} \tag{5}$$

where $v(k+1)|v(k)$ and $u(k+1)|u(k)$ represent the policy prediction for adversaries under the current policy.

2.2 Probabilistic Decision-Making

Consider that pursuers N_2 update the policy $u(k)$ to $u(k+1)$ with a probability defined as

$$p_1(u(k+1)|u(k), v(k), u(k+1) \in \psi_1(z, v(k))),$$

where the policy set $\psi_1(z, v(k)) := \{u, s.t. J(u, v(k)) \leq z\}$, i.e., the policy u ensures that the performance of pursuers remains at or below a specified variable $z \in \mathbb{R}$, given the fixed evaders' policy $v(k)$. Evaders N_2 update the policy $v(k)$ to $v(k+1)$ with a probability defined as

$$p_2(v(k+1)|v(k), u(k), v(k+1) \in \psi_2(z, u(k))),$$

where the policy set $\psi_2(z, u(k)) := \{v, s.t. J(u(k), v) \geq z\}$, i.e.,the policy v ensures that the performance of evaders remains at or above a specified variable $z \in \mathbb{R}$, given the fixed pursuers' policy $u(k)$. Based one-step perception model, the design of policies need to perceive the probability distributions of adversaries' policies, i.e., probabilistic perceptions.

2.3 Problem Formulations

Following the statement in Sect. 1, the risk preference and probability sensitivity can not be neglected in the presence of probabilistic perceptions. Therefore, the significant challenge lies in incorporating subjective irrationality into perception-based MPE differential game and establishing a suitable evaluation framework. Furthermore, the subsequent task of designing the optimal perceived policy presents its own set of complexities. The specific problems can be described as follows:

Problems : Consider MPE differential game subject to (3) and (4), wherein one-step perception model (5) and probabilistic decision model are incorporated. The primary objectives of this paper are twofold:

1) To establish an expected prospect function that effectively captures the risk preference and probability sensitivity in uncertain perceptions;
2) To optimize the complex prospect function with the aim of designing the optimal perceived policy for both pursuers and evaders.

3 The Optimal and Worst Policies Under Deterministic Perceptions

In this section, the optimal and worst policies are designed under deterministic perceptions to provide some convenience for seeking the optimal perceived policy in uncertain perceptions.

Following methods from the theory of optimal control, the optimal policy for pursuers can be designed as $u^*_{min} = -R^{-1}B_1^T P_1(t)z$. Suppose that evaders' policy satisfies $v = S^{-1}B_2\bar{P}_1(t)z$. Then, the matrix P_1 satisfies Riccati equation

$$-\dot{P}_1 = P - P_1B_1^TR^{-1}B_1P_1 - \bar{P}_1B_2^TS^{-1}B_2\bar{P}_1 + 2P_1B_1^TS^{-1}B_2\bar{P}_1. \tag{6}$$

On the other hand, we can design the worst policy for pursuers as $u^*_{max} = R^{-1}B_1^TP_2z$, subject to (3) and

$$J(u^*_{max}, v) = b. \tag{7}$$

Similarly, the optimal policy for evaders can be designed as $v^*_{max} = S^{-1}B_2^TP_3(t)z$. Suppose that pursuers' policy satisfies $u = -R^{-1}B_1^T\bar{P}_3(t)z$. Then, the matrix $P_3(t)$ satisfies Riccati equation

$$-\dot{P}_3 = -P + P_3B_2^TS^{-1}B_2P_3 + \bar{P}_3B_1^TR^{-1}B_1\bar{P}_3 - 2P_3B_2^TR^{-1}B_1\bar{P}_3. \tag{8}$$

Besides, the worst policy can be designed as $v^*_{min} = -S^{-1}B_2^TP_4z$, subject to (3) and

$$J(u, v_{min}) = a. \tag{9}$$

Under deterministic perceptions, the update of policy still depends on the performance function $J(z(0), u, v)$, so the design process of the optimal policy is the same as that without perceptions.

Based on the above statement, the Nash policy (u^*, v^*) satisfies

$$u^* = -R^{-1}B_1^TP_5(t)z, \tag{10}$$

$$v^* = S^{-1}B_2^TP_{15}(t)z, \tag{11}$$