A Pseudo Code of MADRL-STFF

Algorithm 1: MADRL-STFF for Multi-Agent Training

Input: the agent number N, the neighbor agents $\mathcal{N}$ for each agent, state space $\mathcal{S}$, action space $\mathcal{A}$, iteration epoch T, the number of sessions G, discount factor γ, batch size N_{batch}

Output: the parameters of current Q-network f_{θ_π}

begin
  initialize the parameters of current Q-network f_{θ_π};
  initialize the parameters of target Q-network $f_{\theta_{\pi'}}$ by $\theta_{\pi'} \leftarrow \theta_\pi$;
  initialize the capacity of replay buffer $\mathcal{D}$;
  for *episode* $= 1 : G$ **do**
    `/* Lines 6-18: Explore Experience */`
    for *agent* $i = 1 : N$ **do**
      for $t = 1 : T$ **do**
        `/* Lines 8-9: Spatio-Temporal Input Embedding (Section 4.1) */`
        input the state sequence $[s_i^1, ..., s_i^t]$ of agent i into current Q-network f_{θ_π};
        encode the state sequence into feature representations based on Eq. (2);
        `/* Lines 10-12: Spatio-Temporal Feature Fusion (Section 4.2) */`
        fuse temporal features based on Eqs. (3)-(7);
        construct connected subnetworks based on Algorithm 1;
        fuse spatial features based on Eqs. (8)-(14);
        `/* Line 13: Q-Value Prediction (Section 4.3) */`
        predict Q-values for agent i and select action a_i^t by greedy policy;
        calculate the reward r_i^t;
        generate the next state s_i^{t+1};
        store transition $(s_i^t, a_i^t, r_i^t, s_i^{t+1})$ in $\mathcal{D}$;
      end
    end
    `/* Lines 19-21: Update Parameters */`
    sample N_{batch} transitions (s, a, r, s') from $\mathcal{D}$;
    update the parameters of current Q-network based on Eq. (1);
    update the parameters of target Q-network: $\theta_{\pi'} \leftarrow \tau\theta_\pi + (1-\tau)\theta_{\pi'}$;
  end
  return f_{θ_π};
end

B Connected Subnetworks Construction

Algorithm 2: Connected Subnetworks Construction

Input: the agent number N, the neighbor agents $\mathcal{N}$ for each agent, the vehicle queue length L of each agent, the agent number in a connected subnetwork $N_{cs} > 1$

Output: connected subnetworks set CS

begin
  initialize connected subnetworks set CS as empty set;
  for $i = 1 : N$ **do**
    create an empty set cs_i to save agents in the same subnetwork;
    create an empty set NS_i to save the neighbor agents of cs_i;
    add agent i into cs_i;
    set $index = i$;
    for $a = 1 : N_{cs} - 1$ **do**
      add the neighbor agents $\mathcal{N}_{index}$ into NS_i;
      $NS_i = NS_i - cs_i$;
      $index = \arg\min_j |L_i - L_j| \quad j \in NS_i$;
      add agent $index$ into cs_i;
    end
    add set cs_i into CS;
  end
  return CS;
end

C Effect of the Number of Agents in a Connected Subnetwork

In ConFusion, the number of agents N_{cs} in a connected subnetwork influences the effectiveness of our method. To investigate the sensitivity of N_{cs}, the experiments set $N_{cs} = 2, 3, 4, 5, 6$, respectively. The experimental results are shown in Fig. C1.

The experimental results demonstrate that MADRL-STFF achieves the best performance by setting N_{cs} to 4 on all the datasets. In terms of average travel time, the results of MADRL-STFF decrease with the rise of N_{cs} in the beginning. When N_{cs} is larger than 4, the results increase on all the datasets, especially on real-world datasets. The reason may be that the traffic conditions of the intersections in a connected subnetwork are similar when N_{cs} is less than 4. However, with $N_{cs} > 4$, there may be some uncorrelated intersections included in the connected subnetwork. These intersections have large differences in traffic conditions with other intersections in the same connected subnetwork. Therefore, N_{cs} should be set to a proper value to correlate the intersections in the same connected subnetwork. Besides, when $N_{cs} = 4$, the intersections in the connected subnetwork may be distributed around a center intersection. In this situation, the feature fusion in ConFusion is similar to NeiFusion. However, ConFusion can select connected intersections actively, but NeiFusion can only select adjacent

intersections passively. Therefore, ConFusion can pay attention to continuous road sections which may happen traffic congestions.

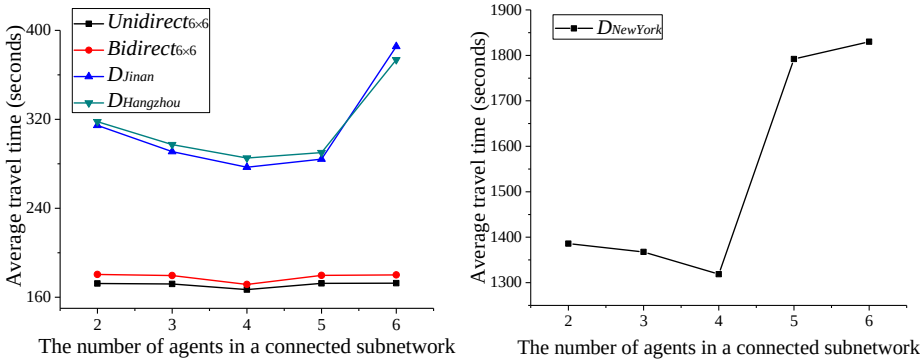

Fig. C1. Effect of the number of agents N_{cs} in a connected subnetwork. (a) The results on $Unidirec_{6\times6}$, $Bidirec_{6\times6}$, D_{Jinan} and $D_{Hangzhou}$ datasets. (b) The results on $D_{NewYork}$ dataset.

D Complexity Analysis

The space and time complexity of MADRL-STFF is analyzed to show the scalability in this part.

For space complexity, if there are L hidden layers in spatio-temporal feature fusion and the hidden layer size is assumed as d, then the size of the weight matrices and bias in each module of MADRL-STFF is: 1) the complexity of spatio-temporal input embedding is $kd+d$; 2) the complexity of spatio-temporal feature fusion is $(d^2+d)L+3M+T+|cs_i|$; 3) the complexity of Q-value prediction is $d|\mathcal{A}| + |\mathcal{A}|$. Therefore, the overall space complexity is $O((d^2 + d)L + 3M + T + N|cs_i| + kd + d + d|\mathcal{A}| + |\mathcal{A}|)$, where d^2 is far larger than the length of the convolutional kernel M and T, the number of agents in a subnetwork $|cs_i|$, the input dimension k and the phase space $|\mathcal{A}|$. Therefore, the complexity of MADRL-STFF is $O(d^2L)$. The space complexity is equal to that of CoLight and STMARL.

For time complexity, there are two assumptions: 1) all the agents predict Q-values concurrently; 2) for one agent, the intersections with its neighbor agents are computed simultaneously. Then the time complexity in each module of MADRL-STFF is: 1) the complexity of spatio-temporal input embedding is Tkd; 2) the complexity of spatio-temporal feature fusion is $T^2d+Td^2L+3M+T+(d^2+d)L+|cs_i|$; 3) the complexity of Q-value prediction is $d|\mathcal{A}|$. Similarly, the complexity of MADRL-STFF is $O(T^2d+Td^2L)$. Compared with STMARL, the self-attention layer in MADRL-STFF costs more time.

E Convergence Analysis

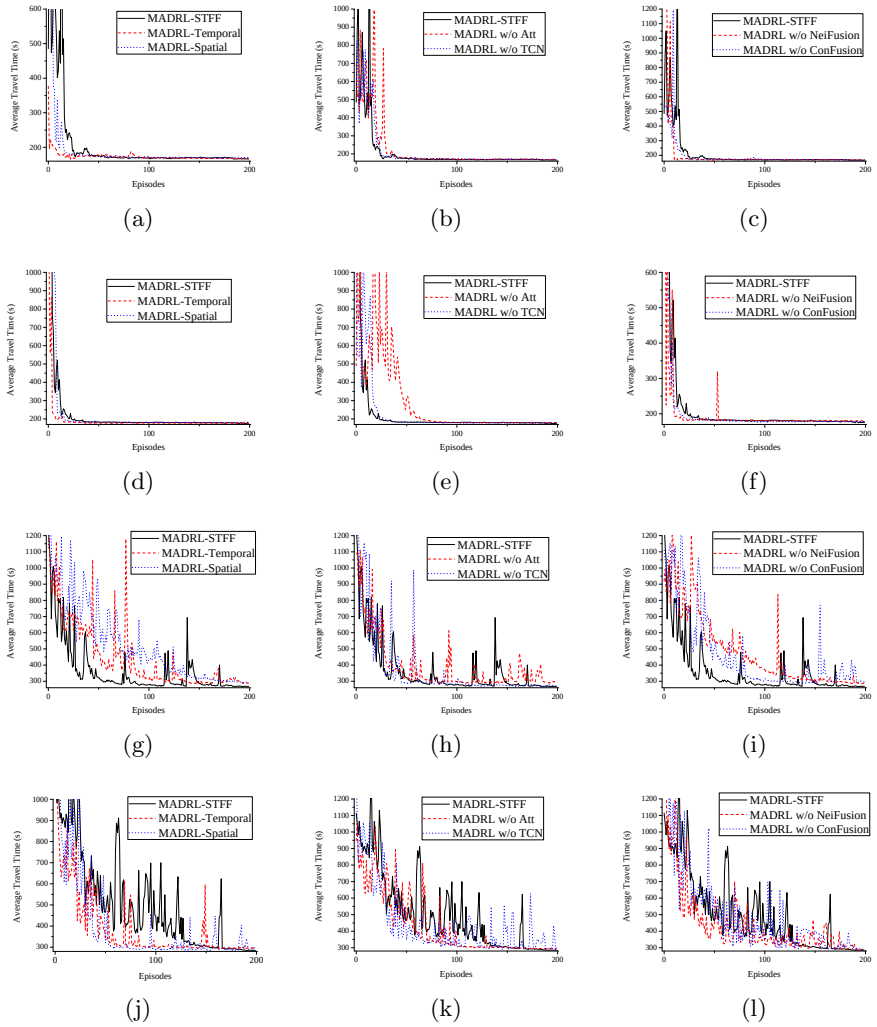

Fig. E1. Convergence curves of MADRL-STFF and its variants in terms of average travel time. (a-c) $Unidirec_{6\times6}$. (d-f) $Bidirec_{6\times6}$. (g-i) D_{Jinan}. (j-l) $D_{Hangzhou}$.

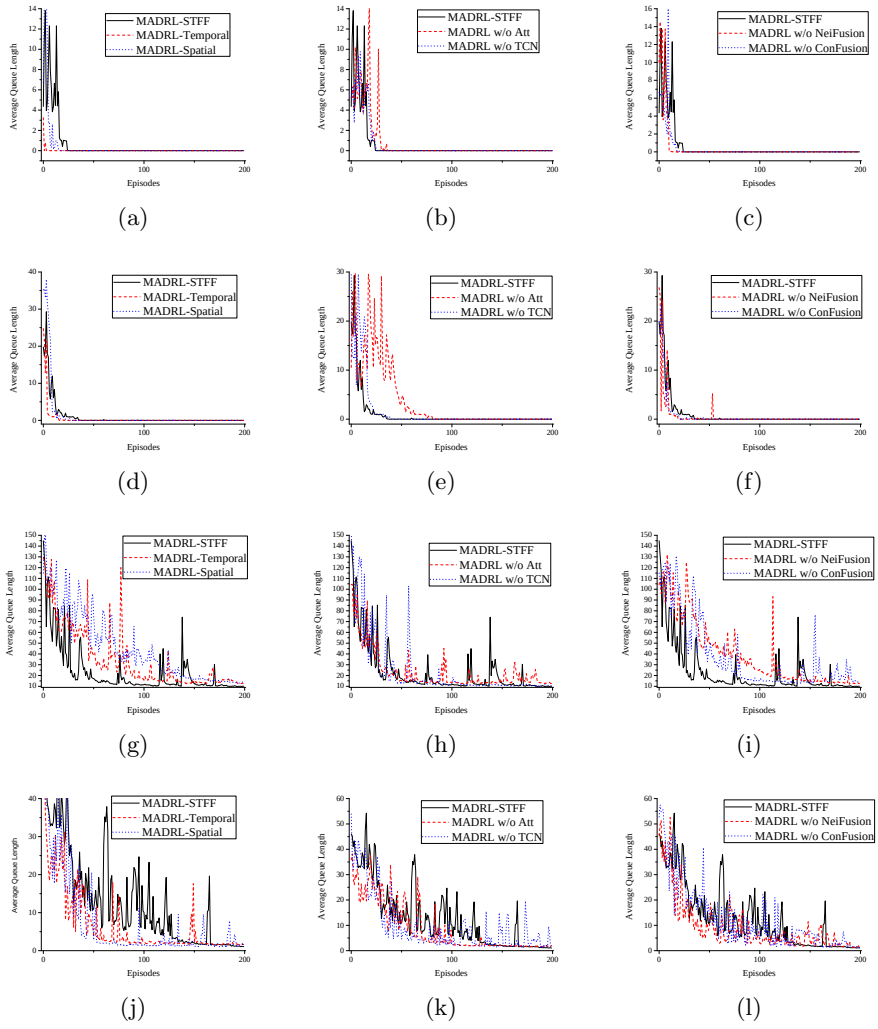

Fig. E2. Convergence curves of MADRL-STFF and its variants in terms of average queue length. (a-c) $Unidirec_{6\times6}$. (d-f) $Bidirec_{6\times6}$. (g-i) D_{Jinan}. (j-l) $D_{Hangzhou}$.

To investigate the convergence speed of MADRL-STFF and its variants, the convergence curves are plotted to show the variation of average travel time and average queue length with the increase of episode. The average travel time is calculated by averaging all the vehicles spent in the roadnet (in seconds). The average queue length is calculated by averaging the rewards of all the agents. On two synthetic datasets and two real-world datasets, the convergence curves in terms of average travel time and average queue length are shown in Fig. E1 and Fig. E2, respectively.

It can be observed that all the methods can converge to stable states, in which MADRL-STFF can converge to the best results in the last several episodes on all the datasets. It demonstrates that MADRL-STFF achieves the better performance than the variants. On synthetic datasets, MADRL-STFF and its variants converge fastly in less than 70 episodes. However, on real-world datasets, most of the methods converge slowly and present waving curves. The reason may be that the traffic flows in synthetic datasets are generated with a regular time interval. Therefore, it is easy to learn the traffic flow patterns. However, on real-world datasets, the vehicle trajectories are complex and the vehicle entering time is hard to predict. Therefore, it costs a long time to learn the traffic flow patterns. Besides, MADRL-STFF converges slower than its variants in some scenarios. It may demonstrate that MADRL-STFF lacks enough coordination among components, which will be improved in the future work.

F Attention Study

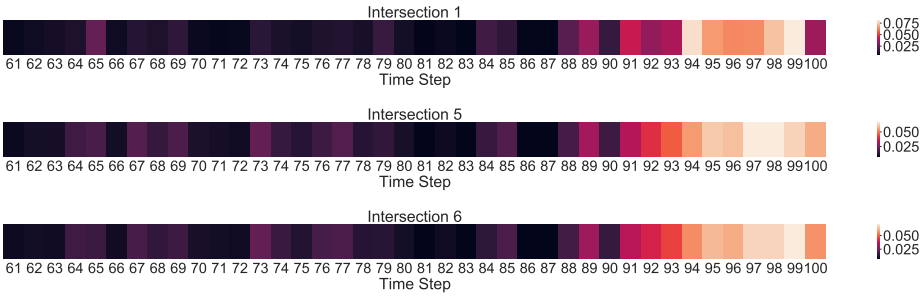

Fig. F1. The attention study of intersection 1, 5 and 6 on $D_{Hangzhou}$ dataset. For the time step 100 in CityFlow simulator, recent 40 time steps are selected to visualize their attention weights.

In temporal feature fusion, a self-attention layer is applied to pay attention to the correlations between two different time steps. Therefore, to investigate the correlations of recent T time steps to the current time step, for the time step 100 in CityFlow simulator on $D_{Hangzhou}$ dataset, recent 40 time steps are selected to visualize their attention weights. The attention visualizations of intersection 1, 5 and 6 on $D_{Hangzhou}$ dataset are shown in Fig. F1. The locations and the states at the time step 100 of intersection 1, 5 and 6 on $D_{Hangzhou}$ are shown in Fig. F2.

The visualization results demonstrate that the features in closer time steps are more helpful for the current time step to select traffic phases. The attention weights in the time steps from 94 to 100 are larger than those in the time steps from 61 to 93. It can be observed that the closer to the current time step, the larger the attention weight. Specifically, for intersection 1, the time step 94 has

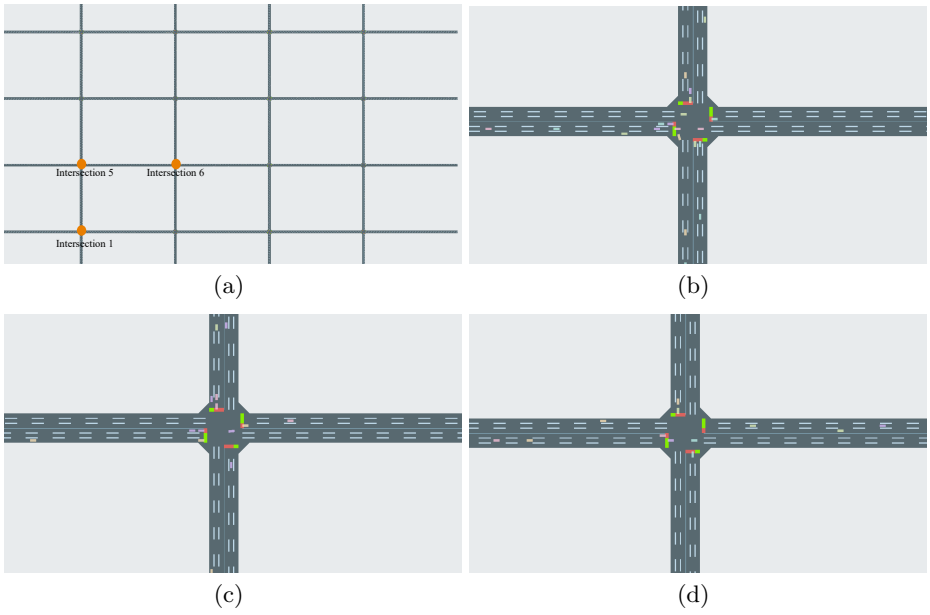

Fig. F2. The locations and the states at the time step 100 of intersection 1, 5 and 6 on $D_{Hangzhou}$. (a) The locations of intersection 1, 5 and 6 in roadnet. (b) The state of intersection 1 at the time step 100. (c) The state of intersection 5 at the time step 100. (d) The state of intersection 6 at the time step 100.

larger attention weight than the time steps from 95 to 98. The reason may be that intersection 1 has more complex traffic condition. More vehicles gather in intersection 1 than intersection 5 and 6 at the time step 100. Therefore, it needs farther time steps to learn the origin of this complex traffic condition for intersection 1.

In spatial feature fusion, NeiFusion fuses the features of neighbor intersections to help the target intersection to make coordinative decisions. To investigate the importance of neighbor intersections to the target intersection, the states of intersection 6 and its neighbors at the time step 100 on $D_{Hangzhou}$ dataset are selected to visualize their attention weights. The attention visualizations are shown in Fig. F3. The states at the time step 100 of the neighbors of intersection 6 on $D_{Hangzhou}$ are shown in Fig. F4.

The visualization results demonstrate that the neighbors with larger traffic flows play more important roles in controlling the traffic signals for the target intersection. For intersection 6, the attention weights of the east, west, north and south intersections are 0.6125, 0.1428, 0.0974 and 0.0556, respectively. The east and west intersections have larger attention weights than the north and south intersections. Meanwhile, it can be observed that the traffic flows in the east and west intersections are more than those in the north and south intersections. It can be verified that intersection 6 includes its east and west neighbors along an

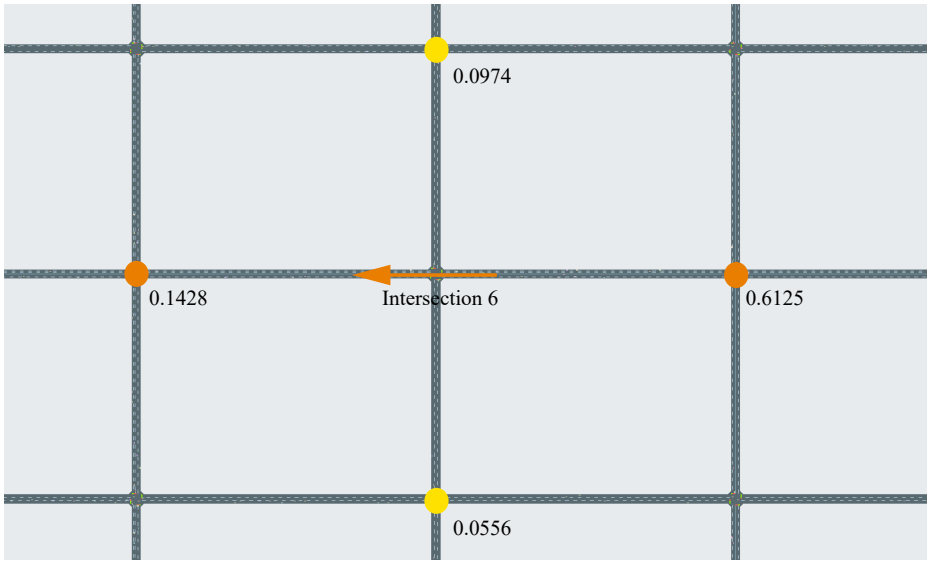

Fig. F3. The attention study of the neighbors of intersection 6 at the time step 100 on $D_{Hangzhou}$ dataset. The attention weights of the east, west, north and south intersections are 0.6125, 0.1428, 0.0974 and 0.0556, respectively.

arterial street named Wener Road in Hangzhou. Therefore, the traffic flows in this arterial street play an more important role on intersection 6 to control the traffic signals.

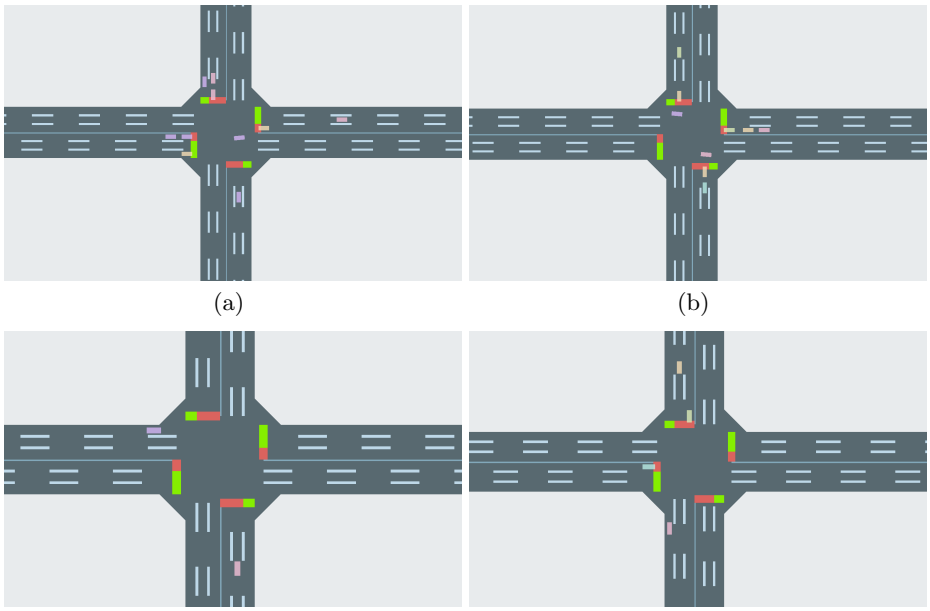

Fig. F4. The states of the neighbors of intersection 6 at the time step 100 on $D_{Hangzhou}$. (a) The west neighbor intersection of intersection 6. (b) The east neighbor intersection of intersection 6. (c) The north neighbor intersection of intersection 6. (d) The south neighbor intersection of intersection 6.