TINET: Learning Invariant Networks via Knowledge Transfer

Chen Luo

Joint work with

Zhengzhang Chen, Lu-an Tang, Anshumali Shrivastava,

Zhichun Li, Haifeng Chen, Jieping Ye

RICE Unconventional Wisdom

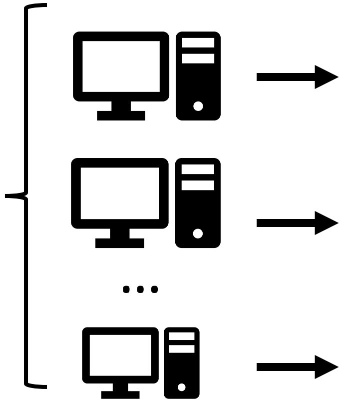
NEC LABORATORIES AMERICA, INC. Relentless passion for innovation

UNIVERSITY OF MICHIGAN

The 2018 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

System Behavior Analysis

Large-scale online services



System Data (Log, Performance Counter, etc.)

System Behavior Analysis

Recommendation
Anomaly Detection
Incident Diagnose
…

Invariant Network Model

- Very Large-scale (More than 80,000 servers for only one data center in AWS)
- 24*7 Running (Online Service)

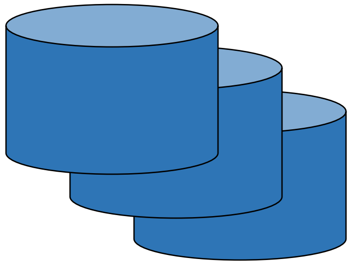
Can not be done manually!

Outline

- Background and Motivation: Invariant Network
- TINET: Learning Invariant Network via Knowledge Transfer
- Experimental Results and Case Study
- Summary

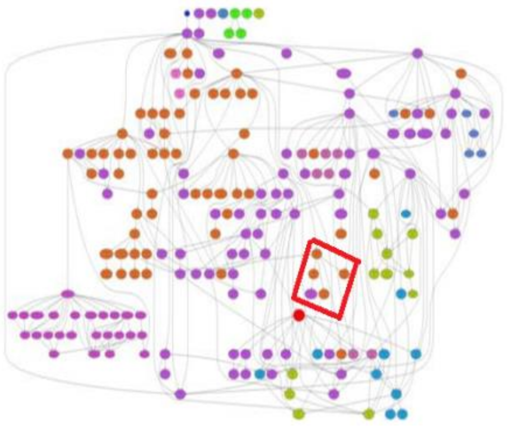
Background: Invariant Network Model

System Data



Construct →

Invariant Networks



Retrieve Model →

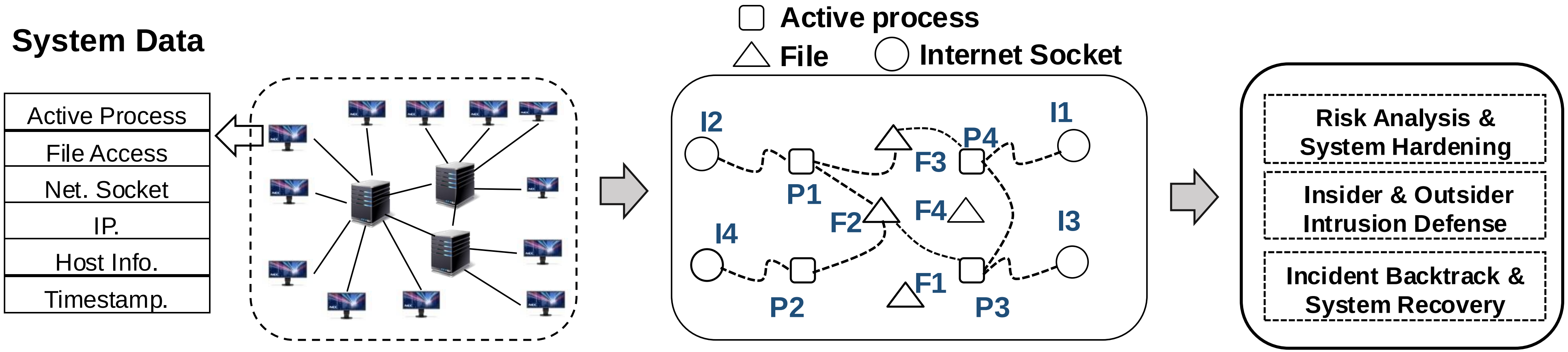
Applications

Recommendation
Anomaly Detection
Incident Diagnose
…

Invariant network captures the **normal behavior profile** of a system.

Invariant network is a powerful and widely-applied tool for further system behavior analysis using **graph mining** algorithms.

Background: How to Build An Invariant Network



(a). Enterprise System (b). Invariant Network (c). Security Applications

Invariant Network is a Heterogeneous Weighted Network:

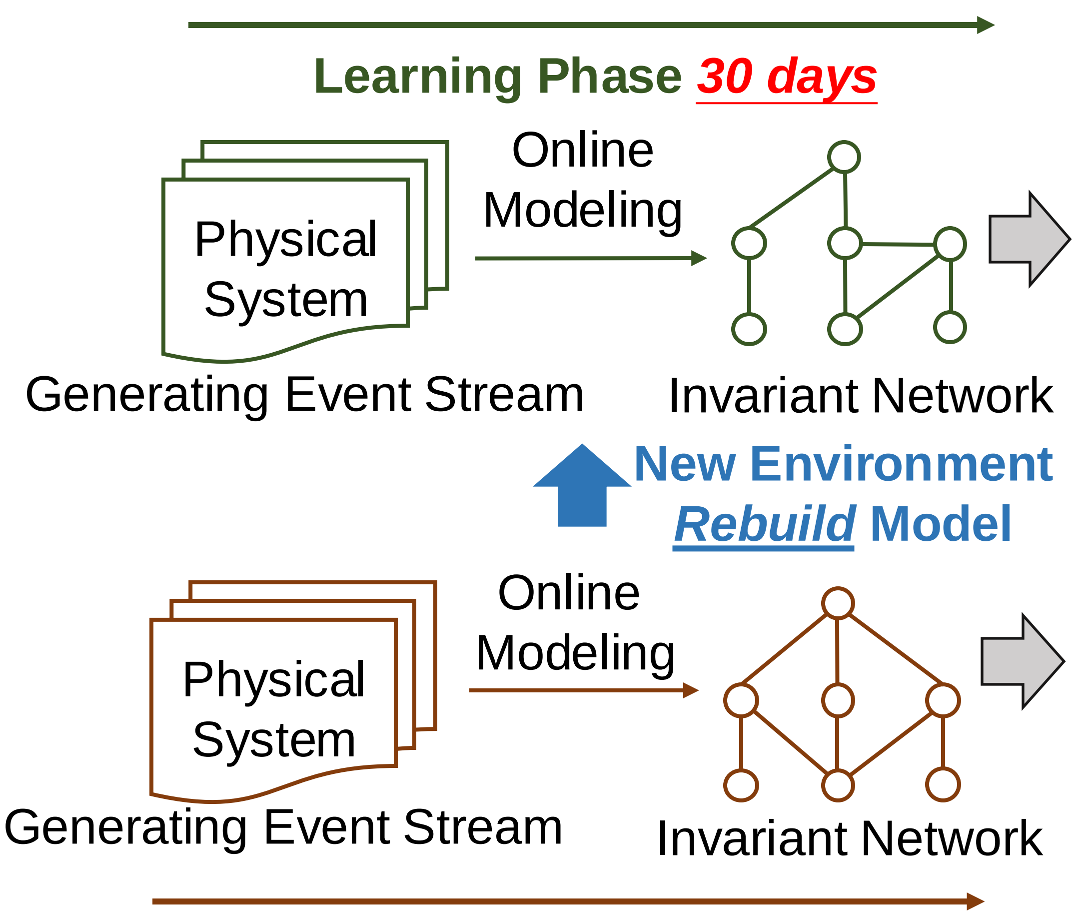
- Node: System component/entity (Process, File, Socket, etc.)
- Edge: Invariant relationship

Constructing Invariant Network

- Time Series data (Cheng, Wei, et al. KDD 2016)
- Categorical data (Boxiang Dong, et al. CIKM 2017)

Motivation: Traditional Workflow of Learning Invariant Network

Traditional Workflow



Applications

- **Recommendation**
- **System Diagnose**
- **Anomaly Detection**
- ……

Issues

1. **Slow:** Constructing a Invariant Network need more than 30 days of learning.
2. **New customer & new environment:** rebuild the network.
3. **POC:** Customer cannot wait.

Motivation: Why Not Directly Transfer Invariant Network?

- Directly transfer the existing invariant network to the new environment
  - Suffered by environment differences **Low stability scores**
  - No knowledge about the new environment **Domain-specific entities or links**

- Existing transfer learning mainly focus on numerical data

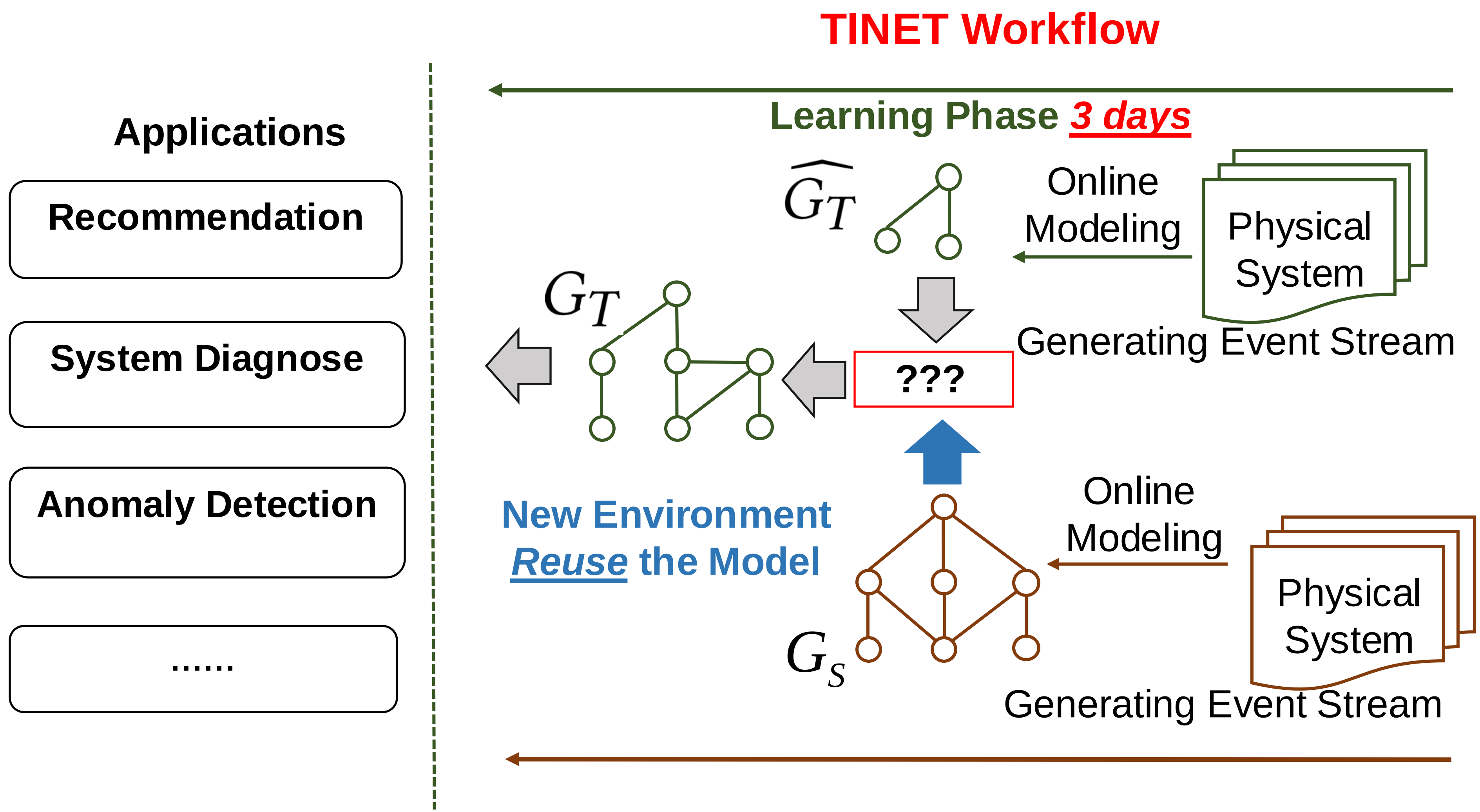
We need a new transfer learning technique for Invariant Network!

In this work, we propose TINET, a knowledge transfer technique for Invariant Networks.

Outline

- Background and Motivation: Invariant Network
- TINET: Learning Invariant Network via Knowledge Transfer
- Experimental Results and Case Study
- Summary

Problem: Knowledge Transfer for Invariant Network

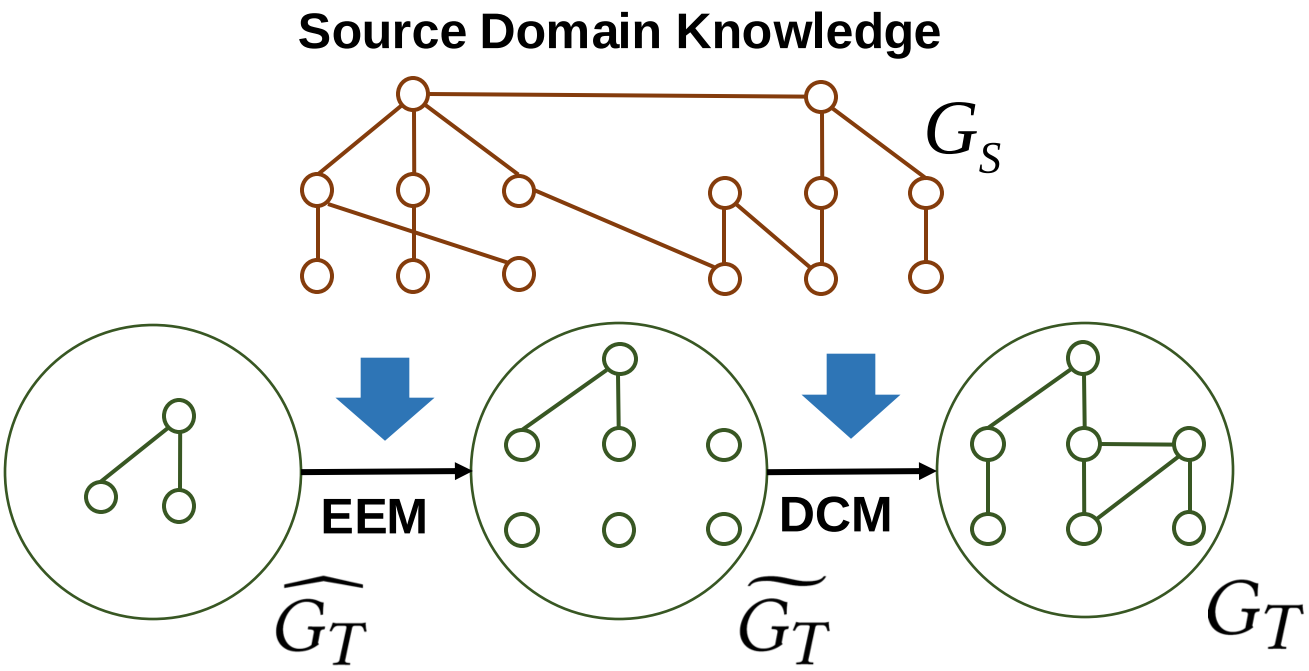


Challenges: Knowledge Transfer for Invariant Networks

- **Extract domain specific knowledge from a target environment.**
  - **The domain specific information is crucial for invariant network learning.**
- **Extract common knowledge from a source environment.**
  - **Only the common knowledge can be transferred from the source domain to the target domain.**
- **How to deal with the heterogeneous relations in the model.**
  - **The network is a heterogeneous graph with multiple types of relations.**

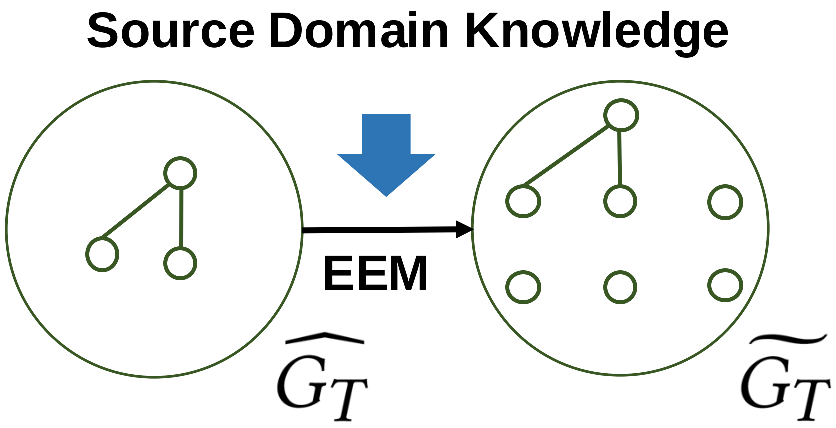
We propose TINET to address all these challenges.

TINET Framework



- **EEM** (Entity Estimation Model)
  - Filter out **irrelevant entities** from source domain
  - Transfer entities to target domain
- **DCM** (Dependency Construction Model)
  - Construct the **missing dependencies** in target domain
  - By solving a **two-constraint optimization** problem

EEM: Entity Estimation Model



- **Two Problems:**
  - Lack of correlation measure for entities
    - **Lean embedding vectors for entities** in a common d-dimensional space
  - Heterogeneous **long-term dependency** relations
    - Meta-paths to represent relations

$$\mathcal{L}_1^{(u_S, W)} = \sum_{i,j}^{n} \left(\| \boxed{u_S(i)} - u_S(j) \|_F^2 - \boxed{S_G} \right)^{\theta} + \Omega(u_S, W)$$

Embedding Term

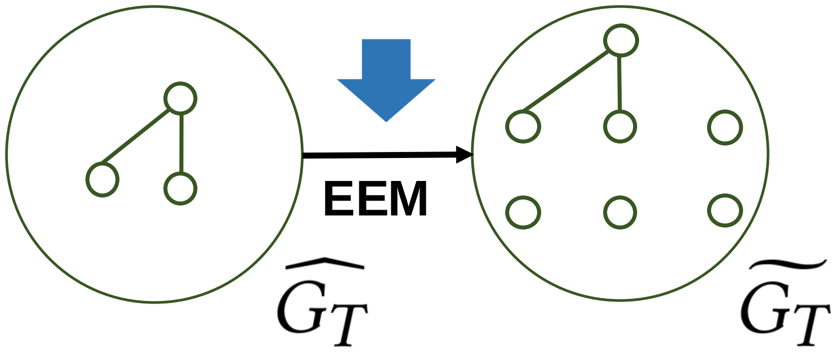
Regularization Term

Embedded vector for each entity

Weighted combination of each meta-path

EEM: Entity Estimation Model

Source Domain Knowledge



Weight for each meta-path

$$S_G = \sum_{i=1}^{|P|} w_i S_{p_i}$$

Adjacency matrix of each homogeneous network

Value equals to shortest path distance

$$\mathcal{L}_1^{(u_S,W)} = \sum_{i,j}^{n} \left(\|\boxed{u_S(i)} - u_S(j)\|_F^2 - \boxed{S_G} \right)^{\theta} + \Omega(u_S, W)$$

Embedding Term

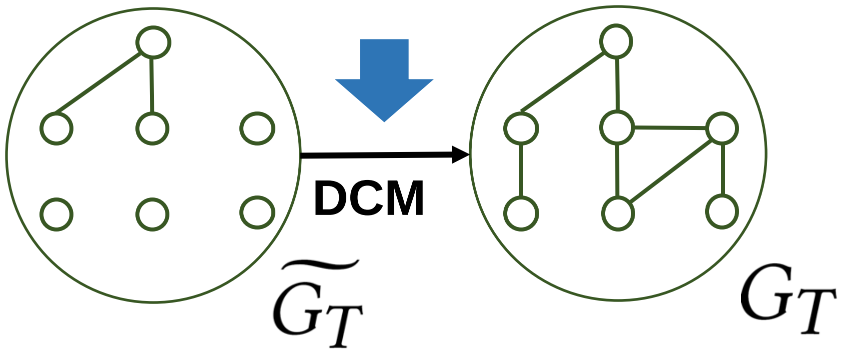
Regularization Term

Embedded vector for each entity

Weighted combination of each meta-path

DCM: Dependency Construction Model

Source Domain Knowledge



- **Constraints**
  - *Smoothness*
    - Learned dependencies should more or less intact in $\widetilde{G_T}$ as much as possible
  - *Consistency*
    - Keep the domain differences

$$\mathcal{L}_2^{u_T} = \mu \mathcal{L}_{2.1}^{u_T} + (1-\mu)\mathcal{L}_{2.2}^{u_T}$$

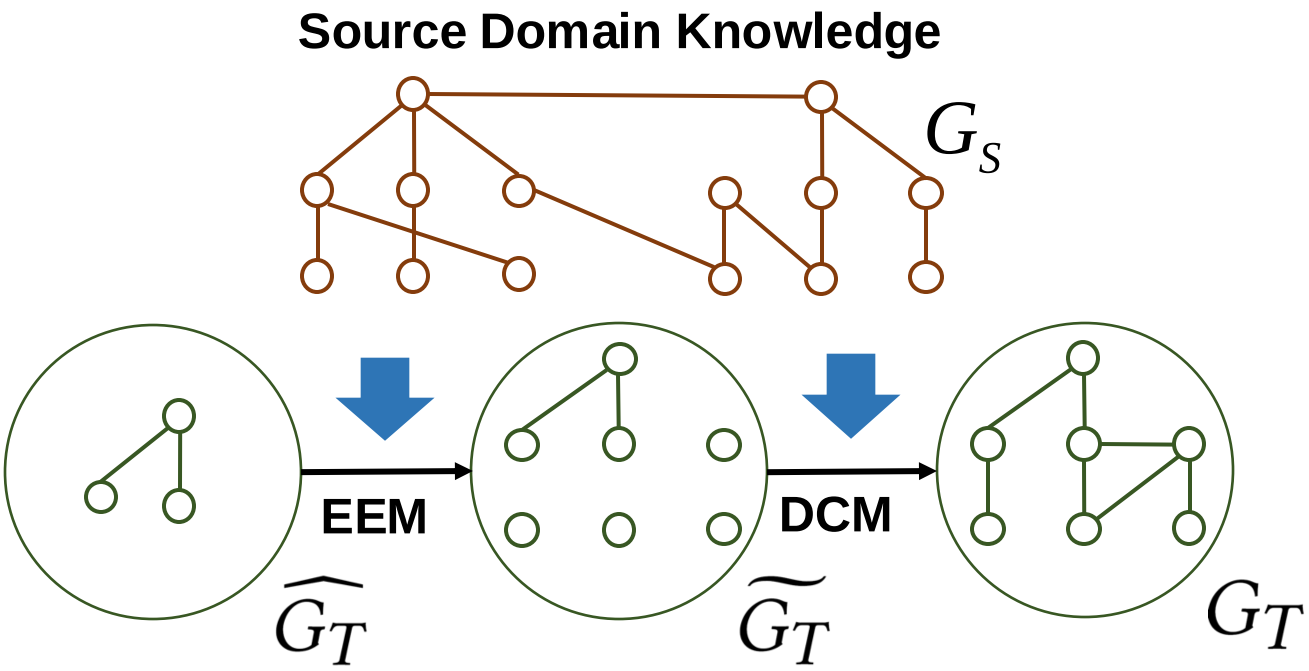
$$= \underbrace{\mu \left\| u_T u_T^T - \widetilde{A_T} \right\|_F^2}_{\text{Smoothness}} + \underbrace{(1-\boxed{\mu}) \left\| \frac{2\left\| u_T u_T^T - \boxed{\widetilde{A_S}} \right\|}{n_S(n_S-1)} - C_3 \right\|_F^2}_{\text{Consistency}} + \Omega(u_T)$$

Matrix representation of source graph.

$$\mu = (|\widetilde{G_T}| - |\widehat{G_T}|)/|\widetilde{G_T}|,$$

Leverage the importance of source information and target information.

Recap: TINET Framework



- **EEM** (Entity Estimation Model)
  - Filter out **irrelevant entities** from source domain
  - Transfer entities to target domain
- **DCM** (Dependency Construction Model)
  - Construct the **missing dependencies** in target domain
  - By solving a **two-constraint optimization** problem

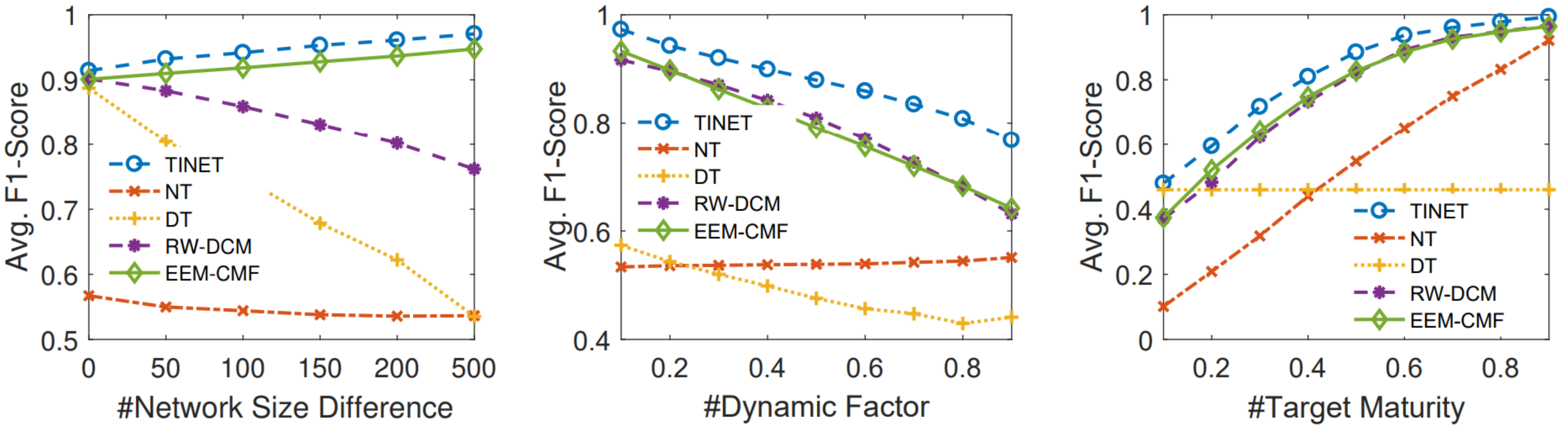
Outline

- **Background and Motivation: Invariant Network**
- **TINET: Learning Invariant Network via Knowledge Transfer**
- **Experimental Results and Case Study**
- **Summary**

Experiment Setup

- **Baselines:**
  - NT: No Transfer (Using the small target graph)
  - DT: Directly Transfer (Using the source graph)
  - RW-DCM: Random Walk + DCM Model
  - EEM-CMF: EEM + Collective Matrix Factorization
- **Datasets:**
  - Synthetic data: generated by three factors (Graph Size, Dynamic Factor, and Maturity Score)
  - Real data: Monitored system data in two OS (Linux and Windows)
- **Evaluation Metrics:**
  - F1-Score: Compare the ground truth Invariant Network with the estimated Invariant Network

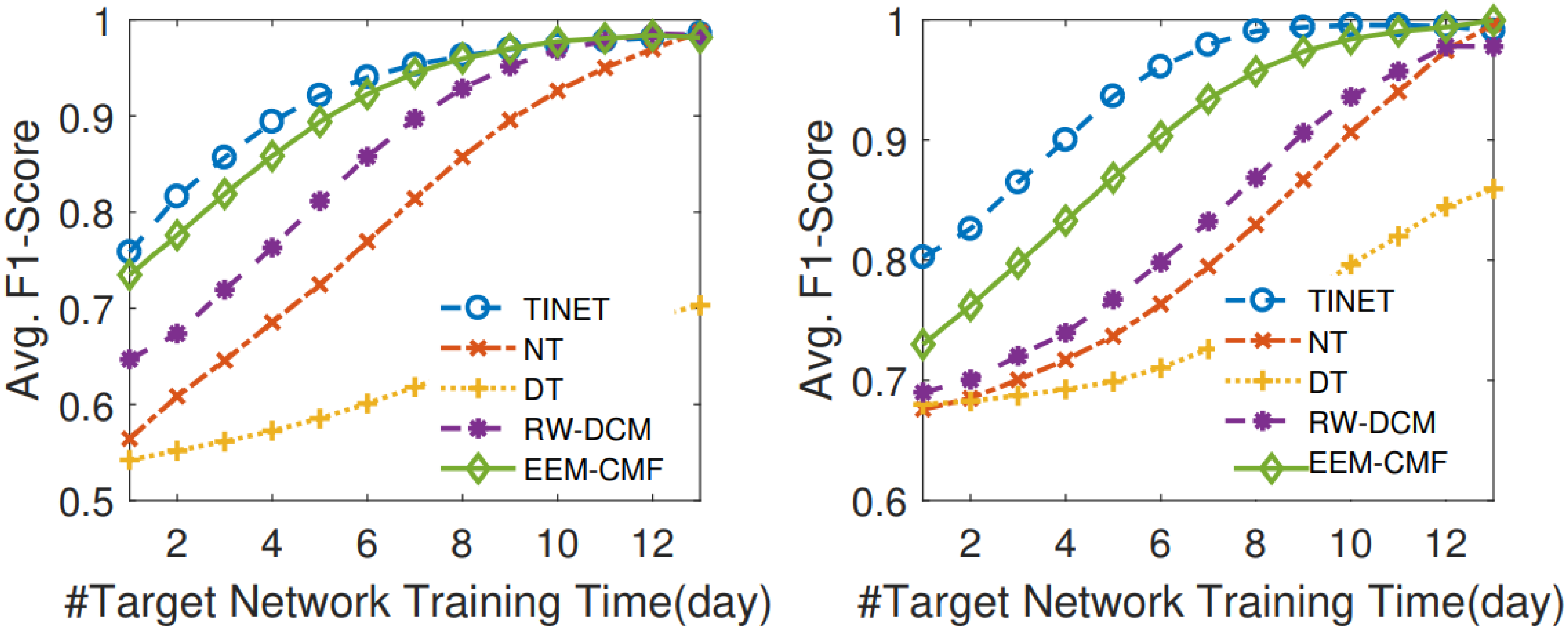
Synthetic Results



(a) Varying graph size **(b) Varying dynamic factor** **(c) Vary graph maturity**

TINET outperforms all baseline methods.

Real Data Results

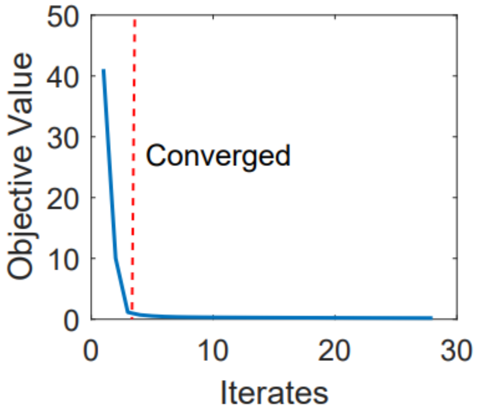


(a) Windows dataset

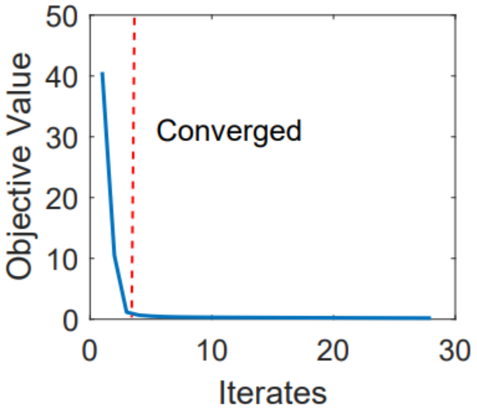
(b) Linux dataset

TINET outperforms all baseline methods.

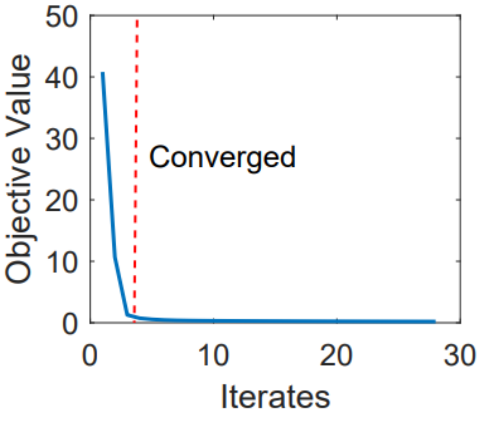
Convergence Analysis



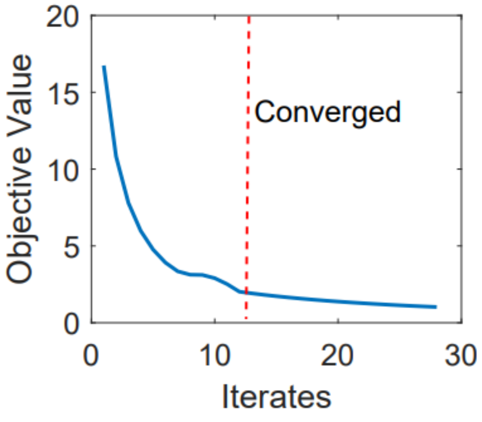
(a) EEM on synthetic data



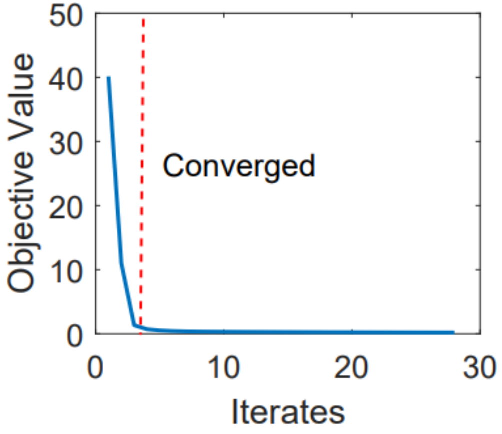
(b) DCM on synthetic data



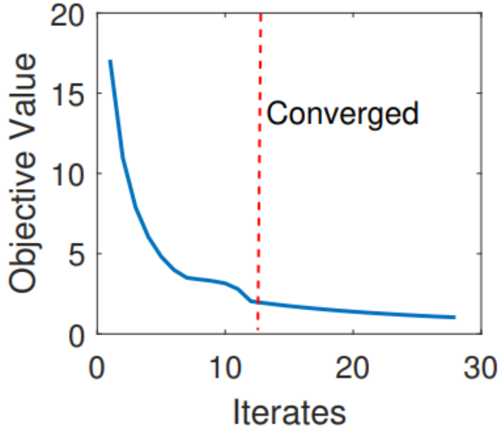
(c) EEM on Windows data



(d) DCM on Windows data

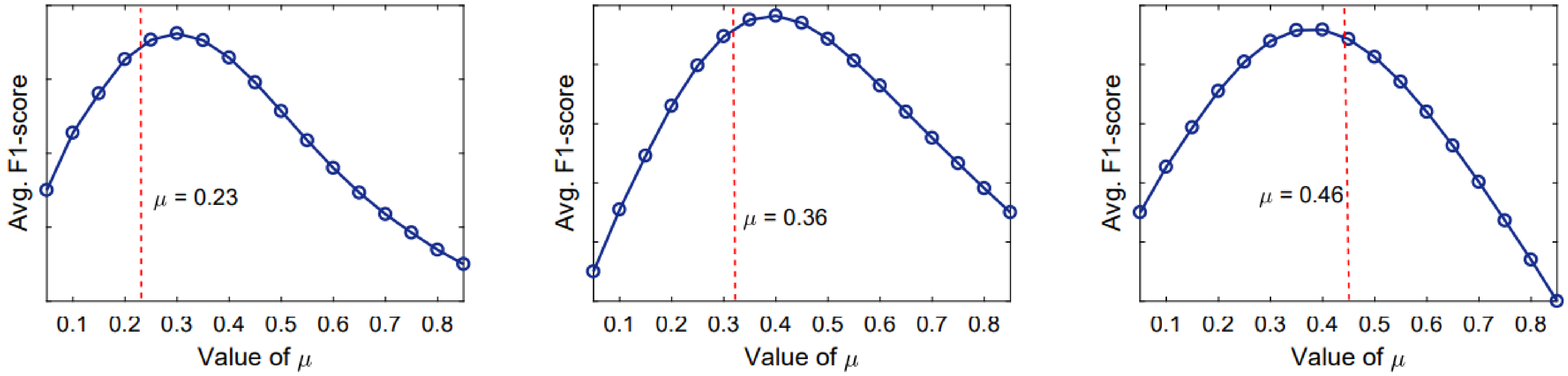


(e) EEM on Linux data



(f) DCM on Linux data

Parameter Study



(a) Synthetic dataset (b) Windows dataset (c) Linux dataset

No parameters needed to be tuned for TINET.

Case Study: Intrusion Detection

Method	Precision	Recall
NT	0.01	0.10
DT	0.15	0.30
RW-DCM	0.48	0.57
EEM-CMF	0.53	0.60
TINET	**0.68**	**0.76**
Real 30 days' invariant network	**0.70**	**0.76**

Source domain: NEC Japan (2 months)

Target domain: NEC Princeton (3 days)

Testing period: 3 days

Launched several cyber attacks for the systems.

$$Recall = \frac{\#Detected\ True\ Alerts}{\#Real\ Alerts}$$

$$Precision = \frac{\#Detected\ True\ Alerts}{\#All\ Alerts}$$

Outline

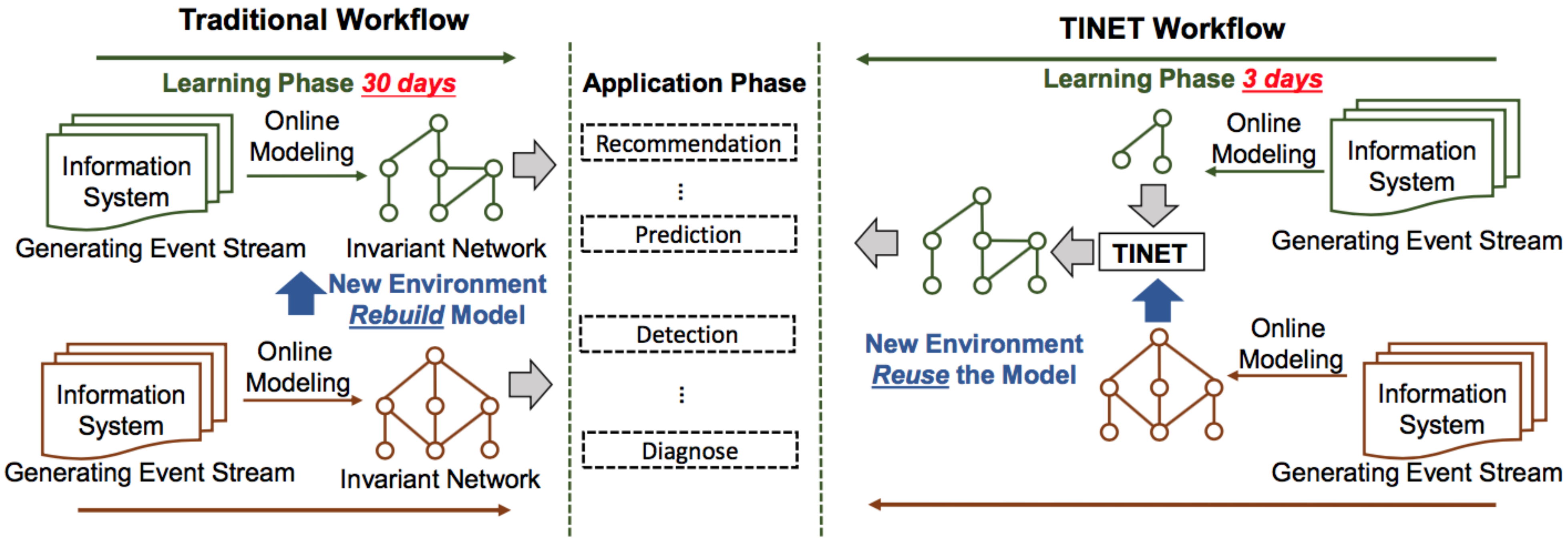
- **Background and Motivation: Invariant Network**
- **TINET: Learning Invariant Network via Knowledge Transfer**
- **Experimental Results and Case Study**
- **Summary**

Summary

- We build the first transfer learning framework TINET for Invariant Networks.
  - TINET can effectively extract useful knowledge from the source domain, and transfer it to the target network.
- We demonstrate the effectiveness of our method on both synthetic and real-world datasets.
  - TINET achieves superior detection performance at least 20 days lead-lag time in advance with very high accuracy.

Thanks

- **QA**



Traditional Workflow
Learning Phase 30 days
Information System
Online Modeling
Generating Event Stream
Invariant Network
New Environment
Rebuild Model
Information System
Online Modeling
Generating Event Stream
Invariant Network
Application Phase
Recommendation
Prediction
Detection
Diagnose
TINET Workflow
Learning Phase 3 days
Online Modeling
Information System
Generating Event Stream
TINET
New Environment
Reuse the Model
Online Modeling
Information System
Generating Event Stream