Network Automation for Path Selection: A New Knowledge Transfer Approach

Guozhi Lin*†, Jingguo Ge*†, Yulei Wu ‡, Hui Li*, Tong Li*†, Wei Mi*† and Yuepeng E*†
*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China
†School of Cyber Security, University of Chinese Academy of Sciences, Beijing, 100049, China
‡College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK
Email: {linguozhi, gejingguo, lihui1, litong, miwei, eyuepeng}@iie.ac.cn, y.l.wu@exeter.ac.uk

***Abstract*—Due to the ever-increasing complexity of modern communication networks, network operators are making tremendous efforts on achieving objectives for the network to meet the diversified requirements of many real-world applications. However, network operators are repeatedly taking a lot of time on some common tasks shared by different networks. In order to reduce repetitive human efforts on network management, advanced machine learning paradigms, such as deep reinforcement learning, has received numerous attention in the networking community. Nevertheless, it encounters great difficulty in transferring learned policies to new environments, resulting in new model training and testing for each changed environment setting. To tackle this important issue, in this paper we propose a new framework that is the first of its kind to enable an agent to have transferable knowledge for network management, specifically, for network path selection tasks. Through this framework, an agent can efficiently learn and express the transferable network knowledge for achieving task objectives. Extensive experimental results show that the learned knowledge through the proposed framework can realize some common objectives of path selection tasks across different network environments. In addition, the knowledge learned from one network task can significantly improve the learning performance of another similar but different task.**

***Index Terms*—Path selection, Transfer learning, Causal learning, Theory-based causal induction, Bayesian inference**

I. Introduction

Network is an important infrastructure for data communication. With the advancement of digital economy and society, the surging number of network data and applications present in modern computer and communication networks. In order to make it stable and functioning in a healthy state, network operators are spending a remarkable amount of time and efforts on network management to achieve some objectives for networks. However, certain objectives are common across different networks, such as shortest path, deadlock free, and minimum delay etc. Operators spend a lot of time to achieve these goals repeatedly in different networks. Manual configurations (e.g., path selection) not only consume tremendous human efforts, but also bring potential risks of network errors. According to a recent Cisco statement [1], 95% of network configuration changes are still performed manually, resulting in much higher operational costs, i.e., 2 – 3 times higher than the cost of the network itself. In order to reduce the risks of network errors and alleviate the repetitive work of operators, it is of paramount importance to develop an intelligent agent to substitute humans for realizing common objectives of network management tasks.

For autonomously achieving objectives, the most important research is reinforcement learning [2] [3]. Its core idea is to learn which action maximizes future reward expectations based on the current state of the environment. It has achieved remarkable results around playing games [4] [5] and developing robot behavior control strategies [6] [7]. Network researchers also used this technique to solve network automation problems, such as virtual network embedding [8], network functions virtulization and placement [9] [10], and resource management in network slicing [11]. Nevertheless, the majority of model-free reinforcement learning methods still have great difficulties in transferring learned policies to new environments [12]. As long as the critical elements in the environment change, it becomes an entirely new environment. Because the policies learned by the model-free reinforcement learning agent are not transferable, the process of learning needs to be repeated to re-learn an effective policy for the new circumstance. As for the network, because the network environment is dynamic and multiform, an agent need be able to correctly achieve objectives under changing environment. In other words, the knowledge that is required to autonomously achieve the objectives of a network task, should be transferable from one network circumstance/setting to another.

In this paper, we take the classic problem of path selection [13] [14], such as routing configuration and load balancing, as an example to study the ability of transferable knowledge for autonomously achieving the objectives of network tasks. Network routing or load balancing protocol is a set of rules that determine how data are transmitted between network nodes. With the diversification of network services, a variety of routing protocols and load balancing strategies have been designed for different application requirements. These protocols or strategies share some common objectives, such as shortest path, deadlock free, and minimum delay. In the following, we use the term *network task* to represent the network management task (e.g., path selection) that has one or more objectives.

Causality is an important factor of knowledge [15] [16]. Through constant interactions and experiments with a physical environment, humans refine the causal hypotheses about the dynamic of the real world. In addition, humans have

ISBN 978-3-903176-39-3© 2021 IFIP

a remarkable ability to transfer causal knowledge between environments governed by the same underlying mechanics [17]. Inspired by the research of autonomous learning in artificial intelligence, the transfer knowledge learning problem can be viewed as a combination of instance-level associative learning and abstract-level causal learning [18]. In this paper, it is the first time that this theory is adapted to the network environment, and based on this adaptation we propose a new hierarchical framework for autonomously achieving the objectives of network tasks. According to theory-based causal induction and Bayesian inference, the proposed framework is devised with three key components: *Abstract-level Causal Structure Learning* representing prior knowledge for a network task; *Instance-level Subconstraints Learning* determining which constraints are associated with a given network task; *Specific Causal Structure Learning* combining the prior knowledge and constraints learned from the network data by virtue of Bayesian inference to obtain the casual structure of the transferable knowledge.

The main contributions of this paper are threefold:

1) We devise a new knowledge hierarchical representation and learning framework for autonomously achieving the objectives of network tasks. Based on the idea of Bayesian inference, the knowledge is divided into prior knowledge and likelihood. In the prior knowledge part, we devise an *atomic schema* and an *abstract schema* to express the abstract casual structure for a network task, called the top-down belief. In the likelihood part, a set of constraints are derived to explain the likelihood probability between the task objective and the network data, called the bottom-up belief. Combining the beliefs of the two parts, an agent is able to effectively learn the knowledge of performing a network task from the network data and encode it into the structural representation that will enable the autonomous achievement of objectives of a network task.
2) In order to verify the effectiveness of the proposed framework and the transferability of the knowledge, we construct a learning environment with a real-world wide area network topology. Running a path selection algorithm on this topology can output the paths between each pair of network nodes. We integrate the top-down belief and the bottom-up belief to produce a highly capable agent that learns the knowledge of generating these paths. Then, we migrate the agent with the learned knowledge into four new environments with different network topologies. By comparing the paths output by the original path selection algorithm and the agent, extensive experimental results show that the agent is able to autonomously complete the path selection task in a new network environment. The learned causal structure knowledge is transferable across different network settings.
3) We also verify whether prior knowledge designed in the proposed framework is beneficial for the agent when learning a different but similar task. We perform two different path selection tasks, one is the task of finding the shortest path, and the other is the task of performing load balancing. After learning the shortest path task, the agent has a prior knowledge of the path selection problem. Then, at the time that the agent performs the load balancing task, we compare two cases in the experiment, one is that the agent uses the prior knowledge learned from the shortest path task, and the other is that the agent does not have such prior knowledge. Experimental results demonstrate that, using prior knowledge, the agent needs less efforts to obtain the causal structure knowledge in most episodes. This proves that the prior knowledge structure designed in the proposed framework has a certain level of universality for learning similar tasks.

The remainder of this paper is organized as follows. Section II presents the problem description. We elaborate the knowledge hierarchical representation and learning framework in Section III. Section IV verifies the effectiveness of the proposed framework. Finally, Section V concludes this paper.

II. The Problem Description

Path selection is a classic routing or traffic engineering problem. For a concrete path selection problem, such as shortest route, network operators have some objectives and code them as technical constraints when searching for a suitable path. The constraint is a set, which includes several subconstraints. Each subconstraint is to achieve a certain objective.

For autonomous network path selection, the purpose of the agent is to learn the structure of constraints, i.e., which subconstraints it includes, and the relationship between them. In the following, we use *causal structure* to represent the structure of constraints. The learned structure forms the knowledge of the agent that can be used to autonomously complete the path selection in any networks.

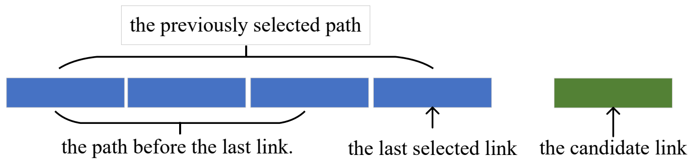
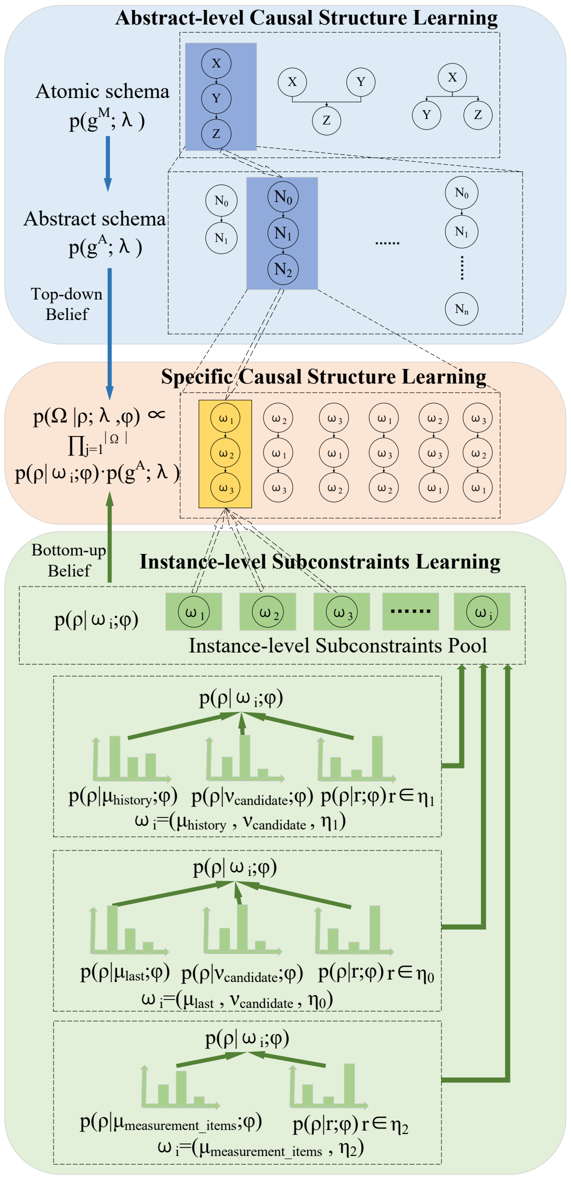


Fig. 1. Schematic diagram

A. Causal structure

The causal structure, denoted as Ω, is formally defined as a set of subconstraints: $\Omega = (\omega_1, \cdots, \omega_k)$, where k is the number of subconstraints ω. In this paper, we enumerate three types of subconstraints: one is the set of restrictions between the candidate link and the last selected link (e.g., they must be connected), another is the set of restrictions between the candidate link and the previously selected path (e.g., in order to avoid loops, the candidate link is not overlapped with the previously selected path), and the last one is the set of restrictions between the candidate link and the network operator's intent on path performance (e.g., the path should be the one with the shortest length or the one with the minimum



Abstract-level Causal Structure Learning
Atomic schema p(g^M; λ)
Abstract schema p(g^A; λ)
Top-down Belief
Specific Causal Structure Learning
p(Ω |ρ; λ ,φ) ∝ ∏j=1^|Ω| p(ρ| ω i;φ)·p(g^A; λ)
Bottom-up Belief
Instance-level Subconstraints Learning
p(ρ| ω i;φ)
Instance-level Subconstraints Pool
p(ρ|μhistory;φ) p(ρ|vcandidate;φ) p(ρ|r;φ) r∈η1
ω i=(μhistory , vcandidate , η1)
p(ρ|μlast;φ) p(ρ|vcandidate;φ) p(ρ|r;φ) r∈η0
ω i=(μlast , vcandidate , η0)
p(ρ|μmeasurement_items;φ) p(ρ|r;φ) r∈η2
ω i=(μmeasurement_items , η2)

Fig. 2. The hierarchical representation and learning framework for transferable knowledge.

delay). To facilitate the understanding of the concepts, we make a schematic diagram as shown in Fig. 1.

The above three types of subconstraints are defined as three tuples, $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$, and $\omega_i = (\mu_{measurement\_items}, \eta_2)$, respectively, where μ_{last} represents the last selected link. A path is composed of links, and $\mu_{history}$ refers to an ordered set of links of a path. $\upsilon_{candidate}$ denotes the candidate link to be selected, and $\mu_{measurement\_items}$ indicates the measurement of link performance. η_0 and η_1 represent the relationship between the first two items of each tuple. η_2 is the requirement of link performance.

The specific combination of these subconstraints constitutes a causal structure. The agent selects the links according to the ω at each time step. For example, $(\mu_{last}, \upsilon_{candidate}, \eta_0)$ may describe that the candidate link must be connected to the previously selected path. $(\mu_{measurement\_items}, \eta_2)$ may illustrate that the next selected link must meet the requirement of the maximum bandwidth. These two subconstraints form a sequential structure, which allows the agent to select the link with the maximum bandwidth connected to the previously selected path.

III. The Hierarchical Representation and Learning Framework

The design principles of the proposed framework come from the theory-based causal induction and Bayesian inference. The former expresses that the induction is the product of domain-general statistical inferences guided by the domain-specific prior knowledge, in the form of an abstract causal theory [19] [20]. Combined with Bayesian inference, it can explain how to learn the knowledge from data at the computational level [21]. For an agent, the hypothetical model of knowledge is regarded as a causal structure. The proposed hierarchical representation and learning framework is shown in Fig. 2. In what follows, we elaborate this framework in terms of framework structure and its associated computation details, respectively.

A. The structure of the framework

The framework consists of three parts from top to bottom, namely, Abstract-level Causal Structure Learning, Specific Causal Structure Learning, and Instance-level Subconstraints Learning, as shown in Fig. 2.

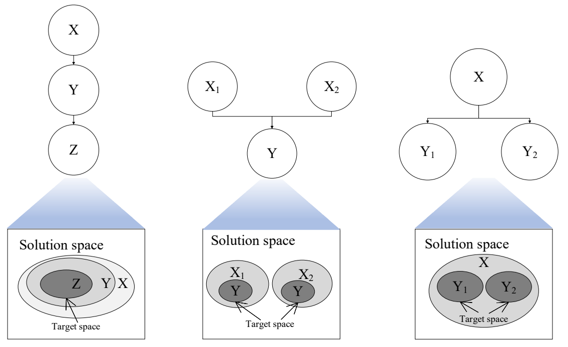
1) Abstract-level Causal Structure Learning: This part follows the assumption that for a given path selection task, all paths form a solution space, and the paths that satisfy the task's objectives form a target space. The solution space contains the target space. The essence that an agent is able to perform autonomous path selection is to use the learned causal structure to obtain the target space from the solution space. It can simply interpret that a causal structure can make a subset of the solution space (subspace). A causal structure consists of several subconstraints. Each subconstraint corresponds to a subspace. The purpose of this part is to provide the abstract causal structures with top-down beliefs about how these subspaces form the target space for the given network task.

We define two parameters, P and R. P attempts to answer the question of what proportion of the target space is among the subspaces confined by subconstraints, and R refers to the question of what percentage of the subspaces derived from subconstraints is among the target space. When P and R are both equal to 1, it shows that the causal structure learned by the agent can accurately obtain the target space. P and R provide a theoretical support and measurement of the rationality of the causal structure. It induces three kinds of basic structural patterns (called atomic schema) of the causal structure, as shown in Fig. 3.

a) Atomic schemas: The atomic schema represents the basic structural pattern of subconstraints in the causal structure. The simplest atomic schema is a linear pattern, which consists of multiple subconstraints connected in tandem, as shown in Fig. 3(a). Note that there is a recursive relationship between X, Y and Z from top to bottom. It means that the top layer subconstraint X is to reduce the solution space of the outermost box. The subconstraint Y of the next layer is to reduce the space of the previous layer subconstraint X. In this way, each subconstraint narrows down the solution space step-by-step and approaches the target space, as illustrated by the lower part of Fig. 3(a).

An effective causal structure needs to meet that the space reduced by the subconstraint of each layer needs to contain the target space completely. This means that the measurement R of X, Y and Z must all be equal to 1. Moreover, the reduced space of the next layer should be closer to the target space, i.e., the measurement P of X, Y and Z should satisfy $P_X < P_Y < P_Z = 1$.

As shown in Fig. 3(b), when X_1 does not include all target spaces, it needs to be supplemented by X_2 in the same layer. Therefore, the multiple-to-one pattern of causal structure is introduced. The above analysis is also applicable to the third pattern: one-to-multiple, as shown in Fig. 3(c)



(a) Linear (b) Multiple-to-one (c) One-to-multiple

Fig. 3. The three types of atomic schemas

b) Abstract schemas: The abstract schema is a set of variants induced by the atomic schema. Each abstract schema can correspond to one of the three atomic schemas by calculating the shortest edit distance [22]. An abstract schema expresses the number and the arrangement of subconstraints provided by a network task, but there is no specific form of instantiating subconstraints. As shown in Fig. 2, in the abstract schema, $N_0, \ldots, N_n$ are placeholders, instead of specific subconstraints. Each placeholder can be instantiated as an arbitrary subconstraint. The purpose of abstract schema is to provide the top-down belief about the abstract causal structure to the Specific Causal Structure Learning part. This belief can be considered as a branch and bound condition to improve the search efficiency when the Specific Causal Structure Learning part searches for a causal structure.

2) Specific Causal Structure Learning: The Specific Causal Structure Learning accepts the top-down belief representing the abstract causal structure and the bottom-up belief denoting the subconstraints learned from the network data, as shown in the orange box of Fig. 2. Combining these two beliefs and using Bayesian inference, the agent makes multiple attempts to learn which combination of subconstraints is the correct causal structure of a network task. When learning is completed, the agent feeds back the correct structural information to Abstract-level Causal Structure Learning so that it can update the belief of the atomic schema and the abstract schema. The detailed computation process of beliefs is elaborated in Section III-B3.

3) Instance-level Subconstraints Learning: In the Instance-level Subconstraints Learning part, the agent learns instance-level subconstraints ω_i from the network data and puts them in an instance-level subconstraints pool, as shown in the green box of Fig. 2. The three green dotted boxes at the bottom represent the likelihood of three types of subconstraints. The subconstraints pool includes all subconstraints and their corresponding likelihood probabilities. The subconstraints in the pool are provided to the Specific Causal Structure Learning part to obtain the causal structure of the network task. The details of calculating instance-level subconstraints ω_i are illustrated in Section III-B1.

B. The computation of beliefs

The belief of the agent comes from two calculation channels of the framework, one is bottom-up belief which is recorded as the φ theory, and the other is top-down belief which is recorded as the λ theory, as shown by the green upward arrow and the blue downward arrow, respectively, on the left-hand side of Fig. 2. These two channels are for computing beliefs and enabling the Specific Causal Structure Learning part to search for the causal structure. In the following, we will elaborate the calculation of bottom-up belief, top-down belief, the integration of these two beliefs, and the top-down belief update.

1) Bottom-up Belief: The agent first collects the bottom-up beliefs from the Instance-level Subconstraints Learning part. As mentioned in Section II-A, the framework has three forms of subconstraints: $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$, and $\omega_i = (\mu_{measurement\_items}, \eta_2)$. There are seven elements, μ_{last}, $\upsilon_{candidate}$, $\mu_{history}$, $\mu_{measurement\_items}$, η_0, η_1 and η_2 that can be considered as random variables, and the agent should learn their probability distribution.

In this paper, we have enumerated several common network attributes, relations and intents. For example, the elements in $\zeta^* = \{\phi_0, \phi_1, \phi_2, \phi_3, \phi_4, \phi_5\}$ can refer to one node connected by a link, the other node connected by the link, the length of the path, link delay, link bandwidth, and link throughput, respectively. We resort to the power set to allow an agent to consider the impact of each combination of attributes on the link selection. If we choose two attributes from ζ^*, i.e., $\zeta_1^* = \{\phi_0, \phi_1\}$, we use $Pow^{\zeta_1^*}$ to denote the power set of these two attributes. The value range of random variables μ_{last}, $\mu_{history}$ and $\upsilon_{candidate}$ is $Pow^{\zeta_1^*}$. The value range of the random variable $\mu_{measurement\_items}$ is the set $\{\phi_2, \phi_3, \phi_4, \phi_5\}$. η_0 and η_1 describe the relationship between μ_{last} and $\upsilon_{candidate}$, and $\mu_{history}$ and $\upsilon_{candidate}$, respectively. In order to make subconstraint $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$ cover more cases, we choose the most common relationship between μ and υ, i.e., equality $\mu = \upsilon$, inequality $\mu \neq \upsilon$, true inclusion $\mu \subset \upsilon$, reverse true inclusion $\mu \supset \upsilon$, and intersection $\mu \cap \upsilon$, denoted by θ_0, θ_1,

θ_2, θ_3, θ_4, respectively. The value range of the random variable η_0 is the set $\{\theta_0, \theta_1, \theta_2, \theta_3, \theta_4\}$. η_1 records the number of times that θ_i occurs in history. Therefore, corresponding to the above five possible relationships, the value range of the random variable η_1 is the set $\{(\theta_0, n), (\theta_1, n), (\theta_2, n), (\theta_3, n), (\theta_4, n)\}$ showing n times occurrence of θ_i in history. In most cases, network management tasks require the network quality-of-service (QoS) to be maximum or minimum in the measurement. For the sake of clarity of illustration, for each measurement item, we consider two possible relationships, i.e., the minimum and maximum values of a measurement item, denoted by θ_5 and θ_6, respectively. Thus, the value range of the random variable η_2 is the set $\{\theta_5, \theta_6\}$.

As for $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, the agent learns a likelihood over which $\mu_{last}, \upsilon_{candidate}$ and η_0 are associated with the instance-level subconstraints that induce the selected paths in the network. In order to reduce the bias introduced by a given distribution, the distribution of μ_{last}, $\upsilon_{candidate}$ and η_0 are modeled as multinomial distributions. Because dirichlet distribution is the conjugate prior of multinomial distribution, the parameters of multinomial distributions can be naturally determined by a dirichlet distribution [23]. There are three dirichlet distributions to maintain the beliefs regarding which combination of subconstraints $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$ is more likely to induce the selected paths. The above also applies to $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$. Thus, there are three dirichlet distributions to maintain the beliefs about which combination of $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$ is part of the causal structure. In $\omega_i = (\mu_{measurement\_items}, \eta_2)$, $\mu_{measurement\_items}$ and η_2 are also encoded as the multinomial distribution. Their parameters are sampled from the dirichlet distribution. The agent maintains two beliefs about $\omega_i = (\mu_{measurement\_items}, \eta_2)$. In summary, there are eight dirichlet distributions to maintain eight beliefs that are involved in the three types of subconstraints, i.e., $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$, and $\omega_i = (\mu_{measurement\_items}, \eta_2)$.

Let ρ represent the selected paths in the network. The agent computes the likelihood of causal structure by decomposing them into subconstraints, as shown by Eq. (1).

$$p(\rho|\Omega;\varphi) = \prod_{\omega_i \in \Omega} p(\rho|\omega_i;\varphi) \tag{1}$$

When the agent learns $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$ and $\omega_i = (\mu_{measurement\_items}, \eta_2)$, $p(\rho|\omega_i;\varphi)$ is formulated as Eqs. (2) – (4) respectively,

$$p(\rho|\omega_i;\varphi) \propto p(\rho|\mu_{last};\varphi)\, p(\rho|\upsilon_{candidate};\varphi)\, p(\rho|\eta_0;\varphi) \tag{2}$$

$$p(\rho|\omega_i;\varphi) \propto p(\rho|\mu_{history};\varphi)\, p(\rho|\upsilon_{candidate};\varphi)\, p(\rho|\eta_1;\varphi) \tag{3}$$

$$p(\rho|\omega_i;\varphi) \propto p(\rho|\mu_{measurement\_items};\varphi)\, p(\rho|\eta_2;\varphi) \tag{4}$$

This instance-level subconstraints likelihood encodes a naive Bayesian prediction of how likely a given subconstraint has occurred in the selected paths in history, without regarding for a task structure. This Instance-level Subconstraints Learning part in Fig. 2 provides the agent with the basic knowledge about which subconstraints are more likely to induce a path.

2) Top-down Belief: The agent learns abstract-level causal structure to encode generalized abstract causal structure about path selections tasks, which is invariant to different network environments. The belief of each atomic schema $p\left(g^M;\lambda\right)$ is modeled as a multinomial distribution, whose parameters are determined by a dirichlet distribution $Dir(\alpha^M)$. g^M refers to an atomic schema, and α^M is the parameter vector whose length is the number of atomic schemas. By sampling from $Dir(\alpha^M)$, the atomic schema g^M_{max} corresponding to the maximum $p\left(g^M;\lambda\right)$ is obtained. g^M_{max} induces a series of isomorphic abstract schemas g^A. The abstract schema represents the number and the structure of subconstraints of the causal structure, regardless of the specific form of subconstraints. Its probability distribution is recorded as $p\left(g^A;\lambda\right)$. The belief of abstract schema is also modeled as a multinomial distribution. By sampling from the corresponding dirichlet distribution $Dir(\alpha^A)$, where α^A is the parameter vector whose length is the number of abstract schemas, the agent gets the multinomial distribution $p\left(g^A;\lambda\right)$. The optimal abstract schema g^A_{max}, whose probability is the largest under $p\left(g^A;\lambda\right)$, can be obtained.

3) Combining Bottom-up Belief and Top-down Belief: In the bottom-up belief, although the agent calculates which subconstraints lead to the selected paths, the agent lacks the logical structures among them. In the top-down belief, g^A_{max} provides the agent with a belief about the most likely abstract causal structure of path selection tasks. When searching for the causal structure Ω, g^A_{max} is used as a benchmark to branch and bound the search. The agent combines the bottom-up belief provided by the instance-level knowledge and the top-down belief given by the abstract causal structure, significantly improving the efficiency of the agent to search in the space of causal structures.

The above process can be formalized as the Bayesian inference, as shown by Eq. (5).

$$p(\Omega|\rho;\lambda,\varphi) \propto \prod_{j=1}^{|\Omega|} p(\rho|\omega_i;\varphi)\, p\left(g^A;\lambda\right) \tag{5}$$

where $p\left(g^A;\lambda\right)$ is top-down belief and $p(\rho|\omega_i;\varphi)$ is bottom-up belief. $|\Omega|$ is the number of subconstraints contained by Ω.

4) Belief Update: When the agent successfully finds the causal structure, it is necessary to update the prior knowledge of the agent, that is, the top-down belief. The way of update is by changing the probability distribution of an atomic schema and an abstract schema. The calculation process is presented as follows.

According to the causal structure that has been searched, the agent traces back to the source to find its isomorphic abstract schema, and then it updates the dirichlet distribution. $p^*\left(g^A;\lambda\right)$ is the updated distribution, and the agent uses it to update the probability distribution of an atomic schema. The agent backtracks to the atomic schema by calculating the formula:

$$p\left(g^M|g^A\right) = \frac{1}{Z} exp(-D(g^M, g^A)) \quad (6)$$

$$Z = \sum_{g^M \in \chi_{g^M}} exp(-D(g^M, g^A)) \quad (7)$$

where χ_{g^M} is the space of atomic schemas and $D(g^M, g^A)$ is computed as the minimal graph edit distance between the atomic schema and the abstract schema. exp is the exponential function based on natural constant e. Through this formula, if the agent finds that the atomic schema g^{M*} is most similar one to the abstract schema structure, it updates the dirichlet distribution.

IV. Experimental Results and Analysis

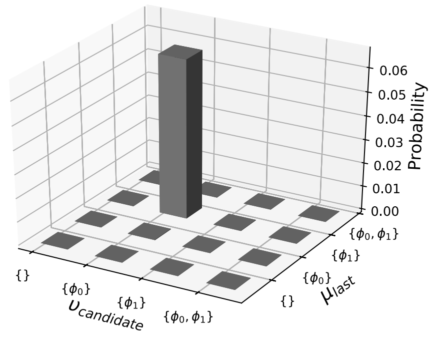
In order to evaluate the effectiveness of the proposed framework for autonomously achieving the objectives for path selection tasks, we conduct extensive experiments to verify the agent's learning performance on subconstraints during the training phase. Then, we migrate the agent to four new networks with different network topologies, and let it find the shortest paths and the load balancing paths between all pairs of nodes to verify the agent's ability of transferring knowledge across different networks. Finally, we compare two experiments, one with top-down belief and the other without top-down belief, to assess the effect of top-down beliefs on the performance of agent's learning ability of the causal structure.

A. Experiments setup

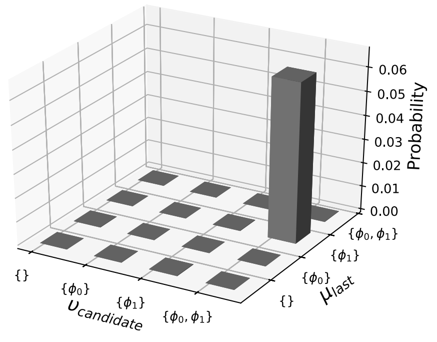
Four network topology attribute files from The Internet Topology Zoo [24] are obtained to validate the effectiveness of the proposed solution. The NetworkX tool takes these files as the input to generate a variety of paths, including those with and without loops. We randomly select some paths and remove their intermediate links to make them disconnected. In this way, we construct our dataset, which includes both the connected and disconnected paths and the paths with and without loops. Besides, we randomly set bandwidth as the weight for each link. The hardware environment of experiments is i5-8265u CPU, 20GB memory and 512GB SSD. The software environment is Ubuntu 18.04, Python 3.7 and NetworkX 2.4.

The experiment is divided into the training phase and the testing phase for evaluating the framework's ability of knowledge transfer for autonomous path selection. In the training phase, we implement the classic shortest path routing algorithm [25] and load balancing algorithm [26] on the network with a given topology to collect the target path between any pair of nodes.[1] The dataset of collected target paths is denoted as D_{target}. At the same time, we implement a common pathfinding algorithm [27] on the network with the same topology to obtain all simple paths between any pair of nodes, denoted by D_{sp}. That is, D_{sp} contains D_{target}, i.e., $D_{target} \subset D_{sp}$. In addition, in order to introduce noisy data to assess the robustness of the proposed framework, we implement a random path selection algorithm and obtain the noisy "path" data, denoted by D_{noise}. The paths in D_{noise} have loops and even false paths composed of disconnected links. In brief, the training process is as follows. First, the agent receives the data of D_{target}, D_{sp} and D_{noise}. Second, it analyzes the characteristics of D_{target} in the background data $D_{sp} \cup D_{noise}$ and infers the subconstraints. Third, it learns the causal structure. This causal structure is the generation model that can generate D_{target} from $D_{sp} \cup D_{noise}$. It is worth noting that the agent does not know the specific algorithm that is used to generate D_{target}. In the testing phase, we migrate the agent to another four new network environments to evaluate the framework's ability of transferring the learned causal structures across different networks.

[1]For the sake of clarity of illustration, we choose classic algorithms as examples for evaluation. It is worth noting that the proposed model framework can be applicable to learn knowledge of any network protocols and algorithms.



(a) Equality $\mu = \upsilon$



(b) True inclusion $\mu \subset \upsilon$

Fig. 4. The agent's learning performance on $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$ with the proposed model.

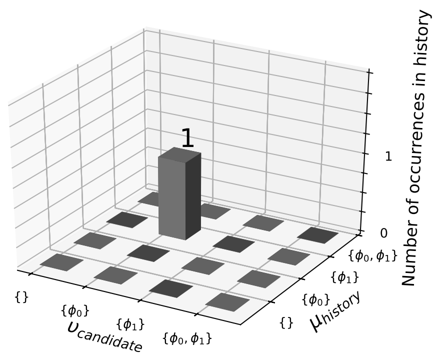


Fig. 5. The agent's learning performance on $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$ with the proposed model, η_1: Equality $\mu = \upsilon$.

B. Learning subconstraints

The results of learning subconstraints $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$, are shown in Fig. 4, demonstrating that the agent is capable of learning the specific combination of subconstraints. The two sub-figures indicate the representations of subconstraints in the two types of η_0 relationships [2], as discussed in Section III-B1, corresponding to the equality $\mu = \upsilon$ and true inclusion $\mu \subset \upsilon$, respectively. As shown in the sub-figure 4(a), the agent learns that the subconstraint $\mu_{last} : \{\phi_1\} = \upsilon_{candidate} : \{\phi_0\}$ shows a high probability in the data. It means that the node ϕ_0 connected by the candidate link $\upsilon_{candidate}$ is the same as the node ϕ_1 connected by the last selected link μ_{last}. The agent encodes the subconstraint $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$ as $\mu_{last} : \{\phi_1\} = \upsilon_{candidate} : \{\phi_0\}$. Take a look at another sub-figure 4(b), the subconstraint $\mu_{last} : \{\phi_1\} \subset \upsilon_{candidate} : \{\phi_0, \phi_1\}$ shows a high probability. It means that one of the two nodes $\{\phi_0, \phi_1\}$ connected by the candidate link $\upsilon_{candidate}$ should be the same as the node $\{\phi_1\}$ connected by the previous link μ_{last}. The agent encodes the subconstraint $\omega_i = (\mu_{last}, \upsilon_{candidate}, \eta_0)$ as $\mu_{last} : \{\phi_1\} \subset \upsilon_{candidate} : \{\phi_0, \phi_1\}$. To facilitate the understanding, its schematic diagram is shown in Fig. 6. Subconstraints with a high probability indicate that the ω_i is learned by the agent. These two sub-figures illustrate the common objectives for path selection tasks: when selecting the next link in the network, the agent must ensure that this link is connected to the previous one. The agent encodes this common-sense knowledge from the perspectives of these two relationships.

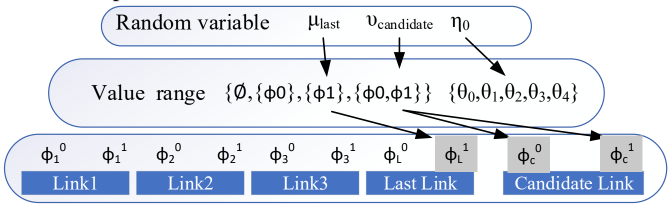


Fig. 6. The sketch map of subconstrain $\mu_{last} : \{\phi_1\} \subset \upsilon_{candidate} : \{\phi_0, \phi_1\}$.

The results of learning performance on $\omega_i = (\mu_{history}, \upsilon_{candidate}, \eta_1)$ are shown in Fig. 5. The prominent column indicates that there is a subconstraint: $\mu_{history} : \{\phi_1\} = \upsilon_{candidate} : \{\phi_0\}, times : 1$. $\{\phi_1\}$ represents the set of nodes connected by the link that the agent has selected before $\mu_{history}$. When the agent selects the next hop link $\upsilon_{candidate}$, the node $\{\phi_0\}$ the link connects to should occur only once in $\{\phi_1\}$. It only occurs once because the next hop link is connected to the last link previously selected, but has no connection to the path before the last link. To facilitate the understanding, its schematic diagram is shown in Fig. 7. It expresses a common object for path selection that the paths do not include loops.

When performing different path selection tasks, network operators have different intentions for the path selection task.

[2]Note, due to space limitation, we only show the experiments for two types, but the experiments for all five types are valid.

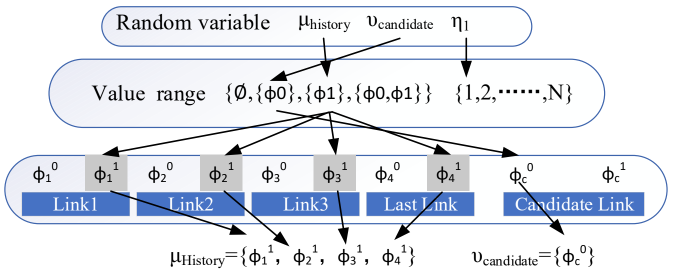


Fig. 7. The sketch map of subconstrain $\mu_{history} : \{\phi_1\} = \upsilon_{candidate} : \{\phi_0\}, times : 1$.

These intentions are usually expressed as pursuing the path to obtain the maximum or minimum value of a certain performance metric. Fig. 8 shows the two intents grasped by the agent. When the agent does not know the specific path selection algorithm, it learns the intention of selecting shortest paths from the network data, and judges that the data are collected from the shortest path algorithm, as shown by the solid line in the figure. In addition, the agent obtains the intention of the path with the minimum delay from the data, and draws the conclusion that the data was collected from the load balancing algorithm. The agent encodes the knowledge about the intent of path selection in the form of probability distribution.

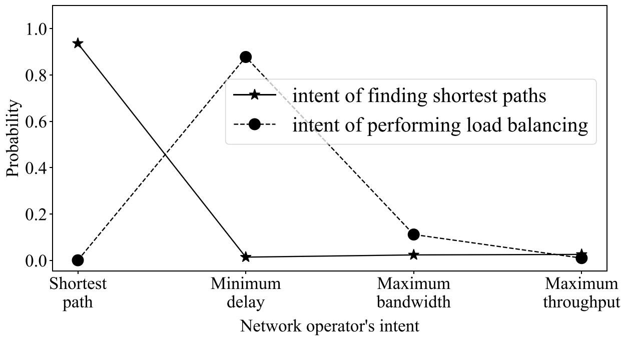


Fig. 8. The agent's learning performance on $\omega_i = (\mu_{measurement\_items}, \eta_2)$ with the proposed model.

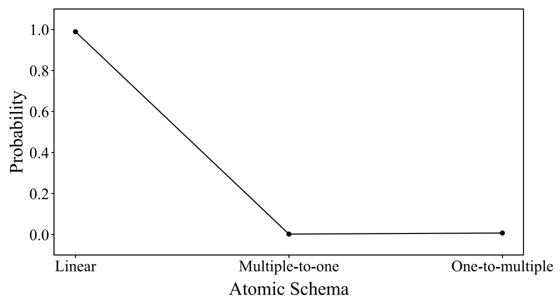
C. Knowledge transfer for the same task across different networks

According to the experimental results conducted in Section IV-B, the common objectives for path selection learned by an agent can be summarized as follows:

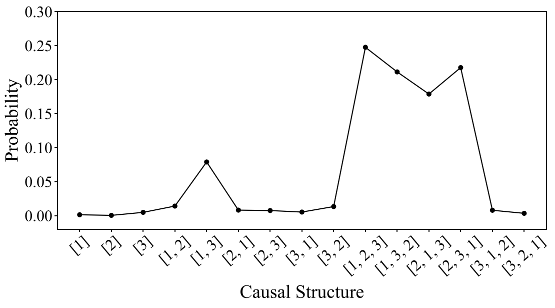
1) When selecting the next link in the network, the agent must ensure that this link is connected to the previous one;
2) The path does not contain loops;
3) The intent of selecting the shortest paths;
4) The intent of selecting the path with the minimum delay.

The above 4 items are recorded as ω_1, ω_2, ω_3, and ω_4, respectively. The agent learns the knowledge of ω_1 and ω_2. ω_3 is learned from the data provided by the shortest path algorithm, and ω_4 is learned from the data provided by the load balancing algorithm.

With ω_1, ω_2 and ω_3, the agent performs 300 random attempts in the experiment. In each attempt, the purpose is to



(a) The agent's atomic schema belief about finding the shortest paths



(b) The agent's abstract schema belief about finding the shortest paths

Fig. 9. The agent's top-down belief of finding the shortest paths.

enable the agent to learn the belief of successfully executing the task of finding the shortest path under different logical relationships of ω_1, ω_2 and ω_3. Fig. 9(a) shows the agent's atomic schema belief about finding the shortest paths. The experimental results illustrate that the linear structure has the highest belief. Under the guidance of this belief, the agent believes that the linear logical relationship is most likely to successfully perform the task of finding the shortest path. Since the linear structure shows the strongest belief, we only reveal the experimental results of the abstract schema induced by the linear structure, as shown in Fig. 9(b). The abscissa in the figure represents the specific causal structure, and the numbers 1, 2, and 3 denote ω_1, ω_2, and ω_3, respectively. Experimental results show that $[1, 2, 3]$ has the biggest belief, which means that in the agent's belief, the causal structure of finding the shortest path task is the sequential structure: $\omega_1 \Rightarrow \omega_2 \Rightarrow \omega_3$.

In order to verify that the knowledge learned by the agent is transferable, we set up four different network topology environments, namely *IBM network topology*, *CWIX network topology*, *BT Europe network topology*, and *Darkstrand network topology*. Fig. 10 shows that the agent uses its belief to execute the causal structure $\omega_1 \Rightarrow \omega_2 \Rightarrow \omega_3$ in four network topological environments to find the shortest path. The abscissa in the figure represents the three steps that the agent completes searching for the shortest path. The first step, ω_1, is to find all the connected paths in the network. The second step, $\omega_1 \Rightarrow \omega_2$, is to remove those paths with loops based on the first step. Finally, in the third step, $\omega_1 \Rightarrow \omega_2 \Rightarrow \omega_3$, the agent selects the shortest path by comparing the length (hops) of the path. Note that in each step, the measurement R is equal to 1, indicating that the agent has not missed any shortest paths in the network. Moreover, the measurement P is improved step by step, demonstrating that the paths selected by the agent at each step are becoming more in line with the requirements of the shortest paths. In the last step, both the R and the P are equal to 1, showing that the agent finds all the shortest paths accurately without any wrong paths.

D. *Knowledge transfer for the different but similar tasks*

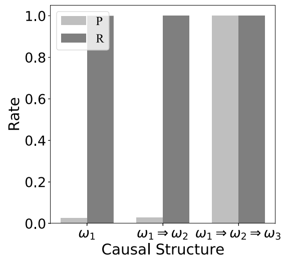
Both the shortest path and the load balancing belong to the path selection problem. The structure of them share a certain similarity. In this section, we will evaluate the effectiveness of our proposed framework on using prior knowledge of performing a network task to improve the performance of learning to carry out a different but similar task. To achieve this purpose, we conduct the following experiments. After learning the shortest path task, the agent has a priori knowledge of the path selection. We make another experiment to verify whether the a priori knowledge learned by the agent from the shortest path task can enable the agent to learn the load balancing task more effectively. When the agent learns the load balancing task, we compare two cases, one is that the agent possesses the prior knowledge learned from the shortest path task, and the other is that the agent does not have such prior knowledge. In Fig. 11, experimental results demonstrate that, using prior knowledge, the agent needs less attempts to obtain the causal structure in most episodes. This proves that the prior knowledge learned through our model has a certain universality for a similar but different task.

E. *Remarks*

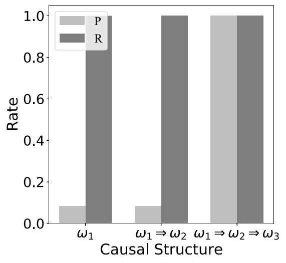
Through the above experiments, it is verified that using the hierarchical architecture we proposed in this paper, the causal structure learned by the agent has the ability of transfer, and the a priori belief of atomic schema and abstract schema is beneficial in the event of learning similar but different tasks for path selection. Using prior knowledge for the discovery of other network management tasks will continue to be studied in our future work.

V. Conclusion

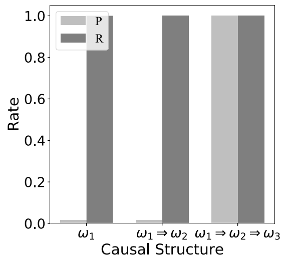
In this paper, we leveraged theory-based causal induction and Bayesian inference to design a new knowledge representation and learning framework for network automation. The proposed framework can enable an agent to have transferable knowledge across different network environments. This framework integrated the top-down prior knowledge and bottom-up likelihood learning. Through our framework, agents can effectively learn and express the transferable network knowledge. Using a classic network path selection task as an example, extensive experimental results showed that the knowledge learned by an agent through the proposed framework can realize autonomous path selection, according to network operator's intents, in different network environments. We also verified through experiments that after learning a network task, the prior knowledge learned is conducive for the agent to learn a similar but different task.



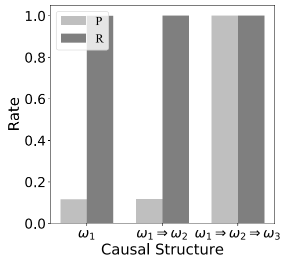
(a) IBM network topology



(b) CWIX network topology



(c) BT Europe network topology



(d) Darkstrand network topology

Fig. 10. Transfer learning performance of finding the shortest paths

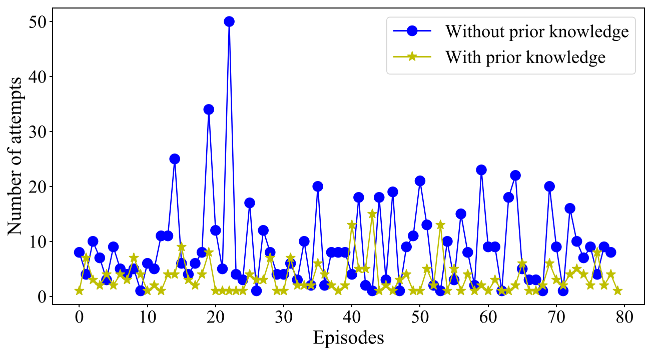


Fig. 11. The performance of the agent when learning load balancing tasks, with and without prior knowledge learned from the shortest path task.

VI. Acknowledgements

This work is supported by the National Key R&D Program of China (Grant No. 2017YFB0801801), Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDC02070200) and the STS (Science and Technology Service Network) Plan of Chinese Academy of Sciences under Grant No. KFJ-STS-QYZD-2021-11-001.

References

[1] What is network automation? Accessed on January 1, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/automation/network-automation.html

[2] R. S. Sutton and A. G. Barto, "Reinforcement learning," *A Bradford Book*, vol. 15, no. 7, pp. 665–685, 1998.

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Articial Intelligence Research*, vol. 4, no. 1, pp. 237–285, 1996.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[6] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[7] T. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2016.

[8] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[9] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 292–303, 2020.

[10] J. S. P. Roig, D. M. G. Estévez, and D. Gündüz, "Management and orchestration of virtual network functions via deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 304–317, 2020.

[11] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2020.

[12] K. Kansky, T. Silver, D. A. Mely, M. Eldawy, M. Lazarogredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," *arXiv: Artificial Intelligence*, 2017.

[13] H. Ahmadi, J. S. Chen, C. S. Chow, R. Guerin, L. Gun, A. M. Lee, and T. E. Tedijanto, "Methods and apparatus for optimum path selection in packet transmission networks," 1993.

[14] F. Kuipers, P. V. Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for qos routing," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 50–55, 2003.

[15] N. R. Bramley, T. Gerstenberg, J. B. Tenenbaum, and T. M. Gureckis, "Intuitive experimentation in the physical world," *Cognitive psychology*, vol. 105, pp. 9–38, 2018.

[16] Y. Zhu, T. Gao, L. Fan, S. Huang, M. Edmonds, H. Liu, F. Gao, C. Zhang, S. Qi, Y. Wu *et al.*, "Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense," *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 1, pp. 27–45, 2020.

[17] M. Edmonds, J. Kubricht, C. Summers, Y. Zhu, B. Rothrock, S.-C. Zhu, and H. Lu, "Human causal transfer: Challenges for deep reinforcement learning." in *CogSci*, 2018.

[18] M. Edmonds, X. Ma, S. Qi, Y. Zhu, H. Lu, and S.-C. Zhu, "Theory-based causal transfer: Integrating instance-level induction and abstract-level structure learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[19] T. L. Griffiths and J. B. Tenenbaum, "Structure and strength in causal induction." *Cognitive Psychology*, vol. 51, no. 4, pp. 334–384, 2005.

[20] ——, "Theory-based causal induction." *Psychological Review*, vol. 116, no. 4, pp. 661–716, 2009.

[21] J. B. Tenenbaum, T. L. Griffiths, and C. Kemp, "Theory-based bayesian models of inductive learning and reasoning." *Trends in Cognitive Sciences*, vol. 10, no. 7, pp. 309–318, 2006.

[22] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image & Vision Computing*, vol. 27, no. 7, pp. 950–959, 2009.

[23] B. A. Frigyik, A. Kapila, and M. R. Gupta, "Introduction to the dirichlet distribution and related processes," *spokesman review*.

[24] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[25] J. Garcia-Luna-Aceves, "A distributed, loop-free, shortest-path routing algorithm," in *IEEE INFOCOM '88,Seventh Annual Joint Conference of the IEEE Computer and Communcations Societies. Networks: Evolution or Revolution?*, 1988, pp. 1125–1137.

[26] K. Gopalan, Tzi-cker Chiueh, and Yow-Jian Lin, "Load balancing routing with bandwidth-delay guarantees," *IEEE Communications Magazine*, vol. 42, no. 6, pp. 108–113, 2004.

[27] J. J. Garcia-Luna-Aceves and S. Murthy, "A path-finding algorithm for loop-free routing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 148–160, 2002.