


RESEARCH

Open Access



Deep bidirectional intelligence: AlphaZero, deep IA-search, deep IA-infer, and TPC causal learning

Lei Xu^{1,2*} 

*Correspondence:

lxu@cs.sjtu.edu.cn;

lxu@cse.cuhk.edu.hk

² Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
Full list of author information is available at the end of the article

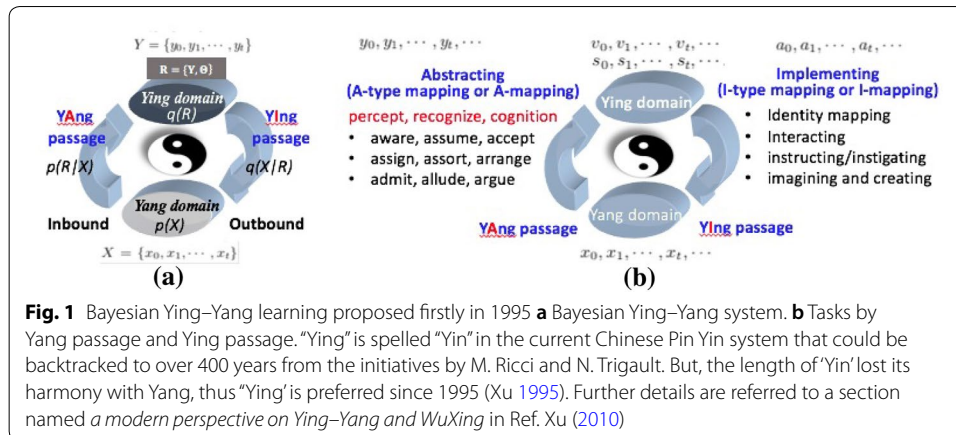
Abstract

This paper starts at a brief review on AlphaGoZero, Q learning, and Monte-Carlo tree search (MCTS), in a comparison with decades ago studies on A* search and CNneim-A that was proposed in 1986 and shares a scouting technique similar to one used in MCTS. Then, we combine the strengths of AlphaGoZero and CNneim-A, resulting in a family named deep IA-search that consists of Deep Scout A*, Deep CNneim-A, Deep Bi-Scout A*, and V-AlphaGoZero, as well as their extensions. Moreover, relation between search and reasoning motivates to extend deep IA-search to Deep IA-Infer for implementing reasoning. Especially, another early study (Xu and Pearl, Structuring causal tree models with continuous variables. In: Proceedings of the 3rd annual conference on uncertainty in artificial intelligence, pp 170–179 1987) on structuring causal tree is developed into a three phase causal learning approach, namely topology identification, parameter reestimation, and causal ρ -tree search on a casual ρ -diagram that is defined by a set of pairwise correlation coefficient ρ . Algorithms are sketched for discovering casual topologies of triplets, stars, and trees, as well as some topologies of casual ρ -Directed Acyclic Graph (DAG), e.g. ones for Yule–Simpson's paradox, Pearl's Sprinkler DAG, and Back door DAG. Furthermore, the classic Boolean SAT problem is extended into one ρ -SAT problem, and the roles of four fundamental mechanisms in an intelligent system are elaborated, with insights on integrating these mechanisms to encode not only variables but also how they are organised, as well as on why deep networks are preferred while extra depth is unnecessary.

Keywords: Deep learning, Deep scouting, Bayesian valuation, MCTS, Path consistency, Star structure, Causal tree, Topology discovery, Conditional independence, ρ -Diagram, ρ -SAT, PROD-SUM, HIERARCHY, Why deep

Background

Recent renaissance of artificial intelligence is basically marked by four types of major advances. First, deep learning and big data achieved high accuracy on recognising patterns, especially human faces and speeches in a huge population. Second, IBM Watson system demonstrated promising capabilities of natural language processing, hypothesis generation, and deep evidence scoring, with successes on conversation system, health-care decision support, contact centre, financial and government services (Ferrucci et al. 2010, 2013). Third, AlphaGo by DeepMind impacted the world firstly by its 4-1 victory



against legendary player Mr. Lee Sedol (Silver et al. 2016) and subsequently by its evolution into AlphaGoZero several months ago that was learnt to perform Go-game simply via self-play, starting from completely random play (Silver et al. 2017). The last but not least, astonishing developments of humanoid robots from ASIMO by HONDA in 2000 to the recent version Atlas by Boston Dynamics, and Sophia by Hanson.

The last three types could be regarded as exemplars of brain-like bi-directional system featured with two complementary subsystems in harmony, while the first one (i.e. deep learning) acts as one key exemplar of inbound subsystem that performs mapping from an external visible domain into an invisible inner domain. The other subsystem performs outbound mapping from invisible inner domain back to the visible observation domain. Recalling Chinese ancient Ying–Yang philosophy and following Bayesian Ying–Yang learning proposed firstly in 1995 (Xu 1995, 1996), as illustrated in Fig. 1a, one subsystem is named Ying machine that consists of an outbound mapping as Ying passage and the invisible domain as Ying domain, while the other subsystem is named Yang machine that consists of an inbound mapping as Yang passage and the visible domain as Yang domain.

Mathematically, the Yang domain X accommodates a set x_0, x_1, \dots, x_t of input data with each x_t in a task-dependent data type, and the Ying domain $R = \{Y, \Theta\}$ accommodates the inner representations of external world, consisting of long-term memory Θ that accommodates model parameters, and short-term memory Y that accommodates the inner representation y_0, y_1, \dots, y_t of x_0, x_1, \dots, x_t . Perceiving, recognising, and cognising via Abstraction, Yang passage $X \rightarrow R$ involves various tasks related words with an initial character “A”, thus also named A-type mapping. Then, thinking process is conducted in the Ying domain and outcomes selected representation $\{Y, \Theta\}$ to reversely drive Ying passage $R \rightarrow X$ that implements various tasks, as illustrated in Fig. 1b, roughly classified into four categories: (1) identity mapping that calibrates whether input can be well reconstructed; (2) interacting with outside (informing, illustrating, communicating etc.); (3) implementing, motoring, instructing, intending; and (4) imagining and creating.

Specifically, it is noted that short-term memory and long-term memory are far from being simply represented by two sets $\{Y, \Theta\}$ in the Ying domain. Elements of both Y and Θ are actually represented in different data types and accommodated in certain procedural and hierarchical structures. In this paper, for example, short-term memory involves an inner state process s_0, s_1, \dots, s_t not only in labels that indicates a trajectory

of concept flow towards goals, but also associated with a flow of attributes $\{y_0, y_1, \dots, y_t\}$ that describe the concepts and instigate not only an action flow a_0, a_1, \dots, a_t to control state transition but also drive the other categories of outcomes. Moreover, s_0, s_1, \dots, s_t is closely coupled with value process v_0, v_1, \dots, v_t that evaluates the prospect of each state towards goals.

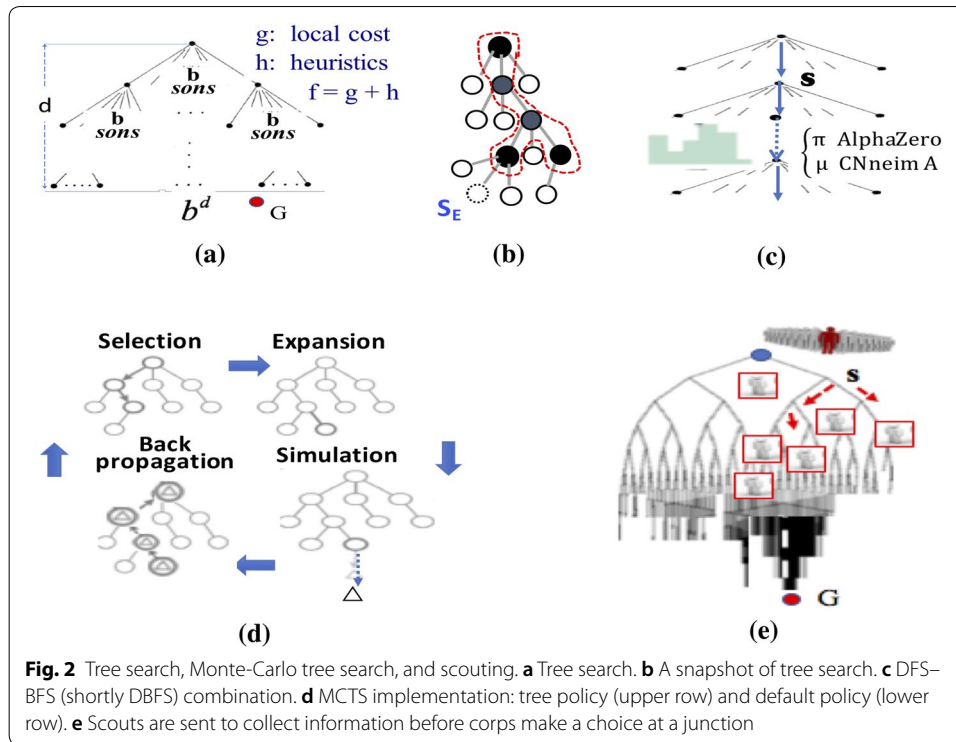
Formulated with help of probability theory, the Ying machine and Yang machine describe the joint distribution of R, X by two kinds of decomposition $q = q(R)q(X|R)$ and $p = p(X)p(R|X)$, respectively. The best harmony learning theory argues that the inner activities R as well as the corresponding $X \rightarrow R$ and $R \rightarrow X$, including updating (or called learning) parameters, are managed by a principle called Ying–Yang harmony maximisation¹ that maximizes $H(p||q) + H(q||p)$, which is in short also referred to as *BYY harmony* or *IA harmony*. From $H(p||q) + H(q||p) = -KL(p||q) - KL(q||p) - E(p) - E(q)$, we see an interpretation that Ying and Yang seek a best agreement via minimising $KL(p||q) + KL(q||p)$ in a vitality or parsimony system via minimising $E(p) + E(q)$.

This paper addresses further possible developments of such brain-like bi-directional systems, with a key nature that deep implementing (or I-type mapping) and deep abstracting (or A-type mapping) in harmony, shortly deep IA harmony or deep Ying–Yang harmony. Examining AlphaGoZero together with revisiting early studies on A^* search, it is interestingly found that MCTS (one key ingredient of AlphaGoZero) actually shares a scouting technique with CNneim-A that was proposed in 1986. Integrating the strengths of AlphaGoZero and CNneim-A, a new method named deep IA-search is proposed, including Deep Scout A^* (DSA), Deep CNneim-A (DCA), Deep Bi-Scout A^* (DBA), and V-AlphaGoZero, as well as their extensions DSA-E, DCA-E, DBA-E, and AlphaGoZero-E.

Considering relation between search and reasoning, we are further motivated to implement reasoning with help of deep IA-search, referred to as Deep IA-Infer. Particularly, casual reasoning are addressed. Another early study (Xu 1986a; Xu and Pearl 1987) on structuring causal tree is developed into a three-phase causal learning approach, namely topology identification, parameter reestimation, and causal ρ -tree search on a casual ρ -diagram that is defined by a set of pairwise correlation coefficient ρ . Also, implementing procedures are further sketched for this TPC causal learning of triplets, stars, and trees, as well as some topologies of casual ρ -Directed Acyclic Graph (DAG), e.g. the ones for Yule–Simpson’s paradox, Pearl’s Sprinkler DAG, and back door DAG.

Moreover, the classic Boolean SAT problem is extended into one ρ -SAT problem, and the roles of four fundamental mechanisms in an intelligent system are elaborated with insights on integrating these mechanisms to encode not only variables but also how they are organised, from which we can interpret why deep networks are preferred while extra depth is unnecessary, and also adopt causal tree or hierarchical SEM equations as an alternative to deep learning.

¹ See $H(p||q) = \int \ln Q d\mathcal{P} = \int p \ln q d\mu$, $KL(p||q) = \int \ln \frac{p}{q} d\mathcal{P} = \int p \ln \frac{p}{q} d\mu$, and $E(p) = - \int \ln \frac{d\mathcal{P}}{d\mu} d\mathcal{P}$.



Tree search and algorithm A*

Problem solving tasks are typically implemented in a large number d of steps. At each step, there are a number b of branches among which we select one to implement, as illustrated in Fig. 2a. The whole procedure is a tree search that starts from the root to a leaf node labeled as a goal G that satisfies a pre-specified condition. Traditionally, there are two classic algorithms for tree search, namely the width-first-search (WFS) and the depth-first-search (DFS), both of which have complexity of the order $O(b^d)$ not only in the worst sense but also in the average sense.

A snapshot of tree search is illustrated in Fig. 2b, which is featured by a red coloured boundary drawn in a dashed closure that divides the tree into two parts. One is the inner part of tree, consisting of nodes inside the closure with each node already expanded and put into a list called CLOSED. The other is the peripheral part of tree, consisting of nodes outside the closure with each node put into a list called OPEN in which all the nodes are expandable but not expanded yet. Made step by step, each step of tree search is expanding one node in OPEN, featured by two jobs. One is usually named *Expansion* that is an action of putting the node S_E that is selected to be expanded next into CLOSE and all its child nodes into OPEN. The crucial point is that each node in either OPEN or CLOSED is associated with one or more attributes to indicate the value of this node. That is, each node should be evaluated before putting into OPEN, for which an appropriate measure is needed and effectively valued as accurately as possible. The other job is named *Selection* that chooses one node from OPEN as S_E . The crucial point is that it is based on a strategy that integrates information not only from different attributes of each node but also from different nodes in OPEN.

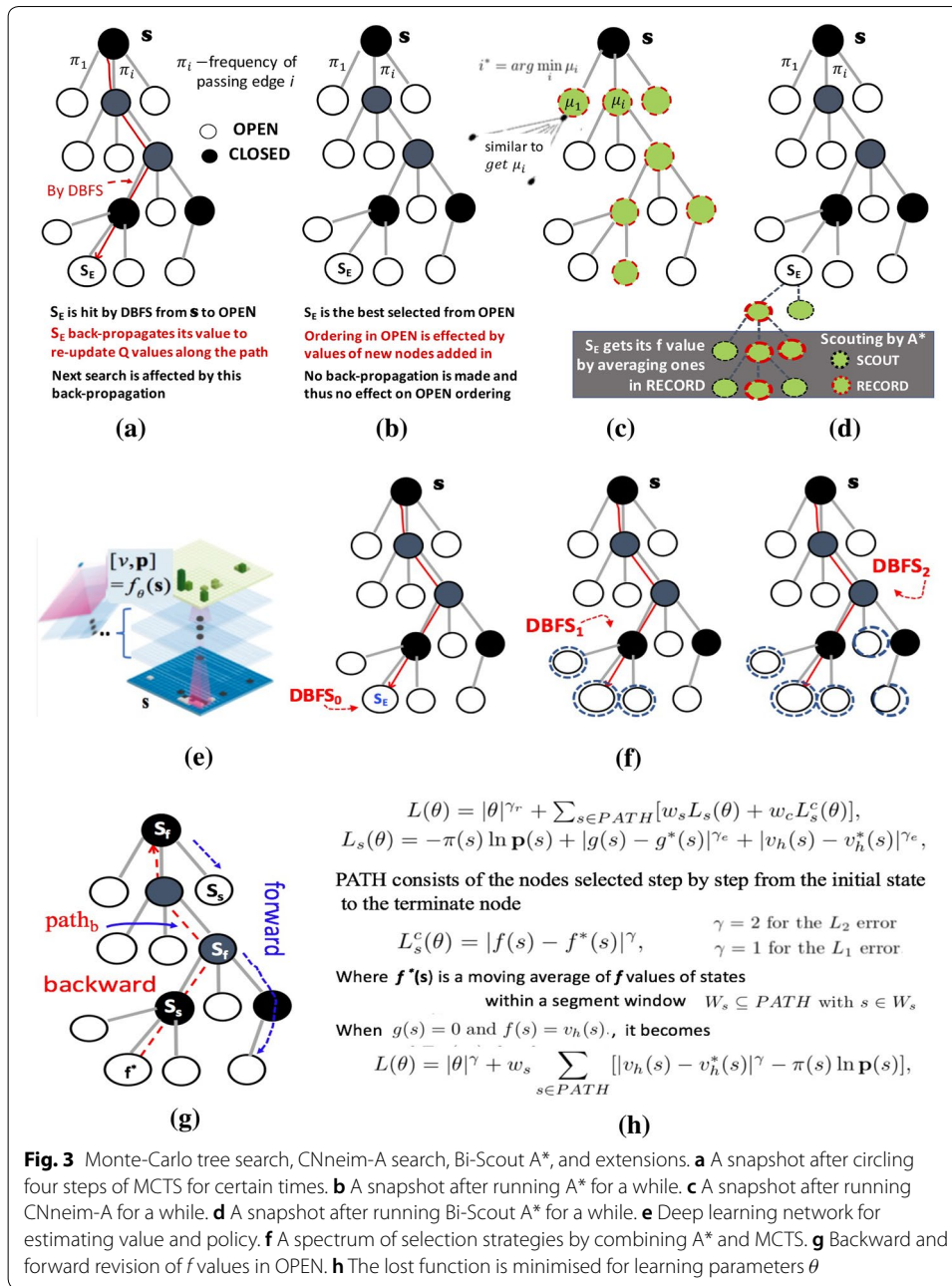


Fig. 3 Monte-Carlo tree search, CNneim-A search, Bi-Scout A*, and extensions. **a** A snapshot after circling four steps of MCTS for certain times. **b** A snapshot after running A* for a while. **c** A snapshot after running CNneim-A for a while. **d** A snapshot after running Bi-Scout A* for a while. **e** Deep learning network for estimating value and policy. **f** A spectrum of selection strategies by combining A* and MCTS. **g** Backward and forward revision of f values in OPEN. **h** The lost function is minimised for learning parameters θ

The simplest case is encountered in WFS and DFS, the value measure is simply an integer ℓ as an attribute to indicate the depth that this node locates at, and the selection strategy is the biggest ℓ for DFS and the smallest ℓ for WFS, respectively. However, this attribute does not directly reflect how good the corresponding node is. Usually, another number f is added as an attribute. The corresponding selection strategy is choosing from OPEN the node with its associated f value being the best among those with all the other nodes in OPEN, e.g. made in those best-first-search (BFS) methods (Xu et al. 1988).

One most popular and classic exemplar of BFS is A* search (Hart et al. 1968; Pearl 1984). As illustrated in Fig. 3b, its selection strategy is selecting s_E to expand as follows:

$$s_E = \arg \min_{s \in \text{OPEN}} f(s), f(s) = g(s) + h(s). \quad (1)$$

where $g(s)$ is known and represents the actual cost from the root to the node s , while $h(s)$ is unknown and supposed to be the optimal cost from s to the goal G . If $h(s)$ could be known, the above s_E would be the best choice and locates on the minimum cost path from the root to the goal G , which would be easily reached by merely d steps. However, $h(s)$ is estimated heuristically (thus named heuristics), which could be poor. Still, the complexity for A^* to find the minimum cost path is typically of the order $O(b^d)$.

There are also examples that combine the uses of the attributes ℓ and f . DFS seeks nodes with the biggest ℓ and quickly goes through a path from the root to a leaf, where the actual cost g^* becomes known and then used as a bound to prune off those branches with its current $g(s) > g^*$. Typical examples are alpha–beta pruning for minimax search (Hart and Edwards 1961) and branch-and-bound technique in many algorithms (Land and Doig 1960). Another example is a combination of DFS and BFS. When DFS tries to pick up nodes with the biggest ℓ , it will encounter more than one nodes associated with the biggest ℓ and simply pick one randomly. Instead, we may use BFS to select the best node according to f value among a subset $\text{OPEN}_{s^*} \subset \text{OPEN}$, where nodes in OPEN_{s^*} associate with the same biggest ℓ and share the same father s^* , resulting in a search path as illustrated in Fig. 2c. In sequel, we will see that such a combination of DFS–BFS (shortly DBFS) leads to different search strategies by different combining ways and specific formulae for f , covering the ones in Q-learning and in MCTS.

Q-learning and reinforcement learning

Considering selection of the best node among OPEN_{s^*} by f given in Eq. (1), we notice that $g(s) = r(s^*, s) + g(s^*)$, $\forall s \in \text{OPEN}_{s^*}$, where $r(s^*, s)$ is an incremental cost or regret locally from $s^* \rightarrow s$, and $g(s^*)$ is constant to every $s \in \text{OPEN}_{s^*}$. Thus, we can modify Eq. (1) into

$$s_E = \arg \min_{s \in \text{OPEN}_{s^*}} Q(s^*, s), Q(s^*, s) = r(s^*, s) + h(s), \quad (2)$$

where $Q(s^*, s)$ represents an estimation of the value from s^* via s to the goal. Setting $h(s) = 0$, Eq. (2) degenerates into a greedy search. This local search is modified by $h(s)$ that estimates a cost from a long term prospect, either improved by a good estimate or deteriorated by a poor estimate.

Known as one most popular reinforcement learning method (Sutton and Barto 1998), Q-learning (Watkins and Dayan 1992) considers the following action policy:

$$a_t = \arg \max_a Q(s^*, a), \quad (3)$$

where $Q(s^*, a)$ represents an estimation of the expected return (reward) associated with taking action a at state s^* . Typically, action a at state s^* leads to one of the child nodes of s^* , that is, $a(s^*) = s \in \text{OPEN}_{s^*}$. In other words, Eqs. (3) and (2) are conceptually the same and act as an example of the previously discussed DBFS search, resulting in a Markov decision process that performs a node-to-node search among OPEN_{s^*} without

memory, and thus is inferior to the search performed by A* in which memory takes role by searching among OPEN.

On the other hand, Q-learning is preferred to A* in that it enhances a winning association pair $\{s^*, a\}$ by updating

$$Q_{t+1}(s^*, a) = (1 - \eta)Q_t(s^*, a) + \eta[r_t + \gamma v(s)], \quad \text{for a step size } \eta > 0, \quad (4)$$

where r_t is the reward received by making action a at s^* and corresponds to $r(s^*, s)$ in Eq. (2), while $v(s) = \max_a Q(s, a)$ corresponds to $h(s)$ in Eq. (2) though it is here discounted by a factor $0 < \gamma < 1$. The difference from either Eqs. (2) or (1) is that a temporal process is considered here because one physical state s^* may be temporally revisited, and thus $Q(s^*, a)$ or $Q(s^*, s)$ is actually a weighted average of its last estimate and a time varying instantaneous estimate $r_t + \gamma v(s)$.

One should not be confused with the notation Q in Eq. (2), where it is used as cost or regret associated with a strategy for cost minimisation and the notation Q in Eq. (3), where it is used as reward associated with a strategy of reward maximisation. Actually, considering either of two directions to optimise, i.e. {cost, min} and {reward, max}, has no fundamental difference and is interchangeable. In sequel, we use Q and also f in Eq. (1) as well without a particular clarification unless it may cause confusion. Also, we use $v_h(s)$ to indicate that either $v(s)$ is considered in reward maximisation or the subscript $h(s)$ is considered in cost minimisation.

Instead of the winner-take-all action policy by Eq. (3), a generic formulation is the following probabilistic policy:

$$p(a|s) = \text{Probability}(\text{taking action } a \mid \text{at state } s), \quad (5)$$

which is sought via optimisation mainly in two ways. One estimates $v_h(s)$ by alternating value iteration and policy iteration, e.g. Eq. (4) is a combined and simplified version. The other is called direct policy search, e.g. deep learning used in AlphaGo (Silver et al. 2016) and AlphaGoZero (Silver et al. 2017). Represented in parametric model $p_\theta(a|s)$, the problem becomes learning θ and is usually handled by a gradient-based search (Sutton et al. 2000).

There are also efforts that apply traditional Monte-Carlo technique to estimate $Q(s, a)$ (Sutton and Barto 1998), still suffering the inferiority of node-to-node search. It is completely different from and should not be confused with Monte-Carlo tree search below.

Monte-Carlo tree search versus CNneim-A search

Monte-Carlo tree search (MCTS) has received remarkable interest due to its spectacular success in computer games (Kocsis and Szepesv 2006; Browne et al. 2012), especially the unbelievable result that AlphaGo impacted the world firstly by its 4-1 victory against Mr. Lee Sedol (Silver et al. 2016).

MCTS methods are featured by circling four steps as illustrated in Fig. 2d for a prefixed large number of times. Instead of giving a conventional introduction, here we provide an alternative insight in a comparison with A* search and together with a further development named CNneim-A.

Illustrated in Fig. 3a is a snapshot after circling four steps for certain times, and illustrated in Fig. 3b is a snapshot after running A* for expanding certain number of nodes.

Being different from A^* that chooses the best node from OPEN according to its associated f value, MCTS starts a new circle at the step of *selection* as illustrated in Fig. 3a for picking one node s_E from OPEN to be expanded next. Search is implemented from the root of tree T in a way similar to DBFS but with f value replaced by $Q(s, a)$, yielding a path that hits one node s_E in OPEN. This node is expanded in the step of *expansion* with all its child nodes put into OPEN, and each child node is valued by the step of *simulation* that runs a default policy to make a fast search, which in the simplest case is to make uniform random moves, until reaching a terminal node that receives a reward value Δ directly from environment. Then, this Δ is delivered up along the path in the step of *back propagation* not only to s_E for updating its v value but also further back to the tree root with Q value on each of its edges updated by Eq. (4), which affects DBFS in the selection step of the next circling.

After a pre-fixed number of circling, the frequency π_i of passing an edge from the root is calculated and used as a probabilistic policy by Eq. (5) to guide a move from s to one of its children, as illustrated in Fig. 2c. One crucial difference between A^* and MCTS is that the search by A^* is made as illustrated in Fig. 3b) without calculation of π_i while MCTS moves as illustrated in Fig. 2c based on π_i that comes from scouting made in Fig. 3a.

Actually, scouting is a strategy that is widely encountered in real life. For example, as illustrated in Fig. 2e, when corps meet a junction, scouts are sent to collect information before making a choice. The strength of MCTS comes from such a scout for computing each π_i . There is a similar scout made in Algorithm CNneim-A that was proposed more than 30 years ago (Xu 1986a; Xu et al. 1987). As illustrated in Fig. 3c, each subtree of the root s is scouted with a pre-fixed number n_i nodes expanded by A^* , obtaining the average μ_i of the f values associated with those expanded nodes to take place of the f value associated with each child i of the root s . Then, a move is made from s to its child according to the best of $\{\mu_i\}$, as illustrated in Fig. 2c.

Apparently, selection by $\{\mu_i\}$ is different from one by $\{\pi_i\}$, but not really. Considering a simple case that there is only one goal G within subtree i from the root s , we randomly assign each node located on the path from $s \rightarrow G$ by 1 with probability π_i and 0 with probability $1 - \pi_i$, while all the other nodes are assigned by 1 with a much smaller probability ϵ and 0 with a probability $1 - \epsilon$. Also, we turn min-selection into max-selection and consider simply DFS by Eq. (3) with tie breaking uniformly. We may roughly observe that μ_i tends π_i as long as n_i is large, and thus the moving policy π may be regarded as merely an extreme case. It may explain why AlphaGoZero needs a temperature parameter τ to adjust (Silver et al. 2017). Containing more information, μ_i already takes such an adjustment in consideration.

Some issues about computing complexity

The idea of scouting subtrees was first appeared in an algorithm named SA^* (Zhang and Zhang 1985). Though SA^* and CNneim-A share a common point of scouting subtrees to collect more information to aid expanding, CNneim-A differs from SA^* critically. CNneim-A simply considers the sample mean of f values on a possible optimal path scouted by A^* in this subtree, justified by the fact that f values on the optimal path should be identical and thus can be regarded as coming from a same distribution.

In contrast, SA^* focuses on Pearl's simplified search space (Pearl 1984; Zhang and Zhang 1985). It is a uniform m -ary tree with one root node and a unique goal node at depth N , where each edge simply has a unit cost 1 and thus we have $g(s) = d$ for a node s at depth d . For a node s in the subtree rooted at the i -th node on the optimal path away from tree's root, we observe $h^*(s) = N - d$ if s is on the optimal path but otherwise $h^*(s) = N + d - 2i$. What SA^* considers is the statistic $a(s) = 0.5[d - N + h(s)]/d$ based on which SPRT test is made to examine whether the population of such $a(s)$ values in a subtree under inspection is significantly different from the population of $a(s)$ values in a subtree that contains the optimal path (Zhang and Zhang 1985).

What can be obtained from such a scouting mechanism? There was a serious confusion in the early theoretical study (Zhang and Zhang 1985). It was surprisingly claimed in Zhang and Zhang (1985) that the mean complexity of order $O(d \ln d)$ was achieved by SA^* search in the same space as Pearl's simplified search space, contradicting to the well known fact that the mean complexity of A^* search is an order growing exponentially with d (Pearl 1984). However, it follows from investigations in Xu (1986a, b, 1987) that this surprising claim unfortunately turned out to be a mistake, due to an incorrect and even contradictory theoretical formulation made in Zhang and Zhang (1985). In Pearl's simplified search space, the fact is that $a(s)$ values actually do not come from a same population, even in a subtree that contains the optimal path.

Nevertheless, the idea of scouting subtrees was newly proposed by that time, being different from those DFS aided lookahead techniques, e.g. alpha-beta pruning (Hart and Edwards 1961) and branch-and-bound (Land and Doig 1960). Sharing the idea of scouting subtrees but implementing selection policy instead of making SPRT test, it was shown by mathematical analysis (Xu 1986a; Xu et al. 1987) that CNneim-A gets the mean complexity of order $O(d^2)$ in general and even of order $O(d \ln d)$ under some constraint in a particularly assumed search space.

Though the search space considered in the above studies is very different from Pearl's simplified search space, the above results do cast insights on those classic results about the mean complexity of A^* search. The scouting-averaging technique used in CNneim-A can improve A^* for two reasons (Xu 1986a; Xu et al. 1987). First, heuristics h becomes easier to estimate for those nodes locate deeper. Second, the variance of the average of a number of random variables is smaller than the variance of each individual random variable, that is, averaging reduces inaccuracy.

Deep reinforcement learning and AlphaGo Zero

No doubt, deep learning (LeCun et al. 2015) and Monte-Carlo tree search (MCTS) (Kocsis and Szepesv 2006; Browne et al. 2012) are two most popular achievements for past decades or more in the AI field. Recently, integration of deep learning, reinforcement learning, and MCTS have outcome not only what called deep reinforcement learning (Mnih et al. 2015; Clark and Storkey 2015), but also AlphaGo (Silver et al. 2016). The key point is using deep network to model a mapping from state configuration s to value function $v_\theta(s)$ and probabilistic policy $p_\rho(a|s)$, such that value and policy estimation are turned into deep learning tasks (Sutton et al. 2000), which enhances both valuing measure and selection strategy in MCTS, as in AlphaGo (Silver et al. 2016) and AlphaGoZero

Table 1 Deep IA-search family

(A)	Deep Scout A* (DSA)	Deep CNneim-A (DCA)	Deep Bi-Scout A* (DBA)	V-AlphaGoZero
Deep learning	$[v_h, g](s) = f_\theta(s)$			$[v_h, p](s) = f_\theta(s)$
Selection step	Get expanding node S_E by A*	In each child tree, get expanding node by A*	Get expanding node by A* subtree scout for μ , by A*	Get expanding node by DBFS
Valuating	Equation (1) by $f = g + h$	Equation (1) by $f = g + h$	Equation (1) with μ replacing f	Q and p by Eq. (6)
Moving policy	Frequency π_i	Mean μ_i	Frequency π_i	Frequency π_i
(B)	DSA-E	DCA-E	DBA-E	AlphaGoZero-E
Deep learning	$[v_h, g, p]_\theta(s) = f_\theta(s)$			
Selection step	Get expanding node S_E by DBFS-n-A* selection			
Bayesian valuation	Make action either stochastically by value q or $\max_a q_a$ upon posteriori $q = [q_a], q_a = p_a e_a / p^T E, E = [e_a]$ $TypeQ : e_a = \rho(Q(s, s_a))$ or $e_a = \rho(r + v_h(s_a))$, where $s_a = a(s)$ $TypeF : e_a = \rho(f(s_a))$, where $s_a = a(s)$			
	$TypeQ : e_a = \rho(Q(s, s_a))$ or $e_a = \rho(r + v_h(s_a))$, where $s_a = a(s)$ $TypeF : e_a = \rho(f(s_a))$, where $s_a = a(s)$			
	If q_a is larger than a pre-specified threshold, put s_a into OPEN, otherwise into WAIT. When OPEN becomes empty, move some ones from WAIT to OPEN. Note: $p(r)$ is monotonically increasing for reward maximisation or decreasing for cost minimisation			
OPEN revision	Revise f values in OPEN by back-forward propagation after each expanding			
Others	Same as DSA	Same as DCA	Same as DBA	Same as AlphaGoZero

(Silver et al. 2017). We focuses on AlphaGoZero since it is the latest and outperforms the former significantly.

AlphaGoZero makes a move from s to one of its children by using π as illustrated in Fig. 2c, while π is calculated from scouting by MCTS, as illustrated in Fig. 3a. A new circle starts at the step of *selection*, where $Q(s, a)$ is not only computed by a simplified version of Eq. (4) with a special setting $r_t = 0, \gamma = 1$ and with $v = v_h = v_\theta(s)$ by deep learning, but also combined with $p(s, a) = p_\rho(a|s)$ by Eq. (5), resulting in $Q(s, a)$ as follows:²

$$Q(s, a) = Q(s, a) + w_q(1 + N(s, a))^{-1}p(s, a), \quad (6)$$

where $N(s, a)$ accumulated the visit count of the edge (s, a) during expanding the present tree T by DBFS. The role of $p(s, a)$ is providing a priori probability for action $s \rightarrow a$, which is discouraged by the number that this action was made. This part is further weighted by a pre-specified coefficient $w_q > 0$. Moreover, as illustrated in Fig. 3e, a deep network with many convolutional layers maps the current state s (i.e. the current configuration of the chess board) into a pair $\{v, p\}$ with $v = v_h = v_\theta(s)$ and $p = p(s, a) = p_\rho(a|s)$.

Running over a pre-fixed number of circling, π is estimated by $\pi \propto N(s, a)^{1/\tau}$, representing the frequency of passing each edge from the root s , where $\tau > 0$ is a controlling temperature. Moreover, θ is updated by stochastic gradient learning to minimise a given

² This one was used in Ref. Silver et al. (2016) and is kept to be same conceptually in Ref. Silver et al. (2017), though an additional factor is multiplied to w_q but may be absorbed into the notation of w_q .

loss function L , once the search guided by π and illustrated in Fig. 2c eventually reaches a game-over state with an indicator z received.

Method

Deep IA-search family

With help of deep learning (DL) network for estimating $[v_h, g](s) = f_\theta(s)$, we can extend A* search and CNneim-A into three DL-based tree search techniques summarised in Table 1(A), as variants or counterparts of AlphaGoZero.

The first is named Deep Scout A* or shortly DSA. We make A* search to expand a search tree T as illustrated in Fig. 3b as the counterpart of the MCTS tree shown in Fig. 3a. As tree expanding process satisfies certain preset condition (e.g. a fixed number of leaf nodes or depth, etc.), we count the number n_i of nodes in the i -th child tree of the root s and make a move based on $\pi_i = n_i / \sum_i n_i$ as illustrated in Fig. 2c, in a way similar to AlphaGoZero but different in getting the tree by A* instead of MCTS.

The second is named Deep CNneim-A or shortly DCA. As illustrated in Fig. 3c, each child tree of the root s is scouted by A* by expanding a pre-specified number of nodes, obtaining the average μ_i of the f values on the longest path from s to a leaf. Then, a move is made from s to its child according to the best of $\{\mu_i\}$ instead of the best of $\{\pi_i\}$ used by AlphaGoZero, as illustrated in Fig. 2c.

The third is named Deep Bi-Scout A* or shortly DBA, which combines DSA and DCA, featured by two levels of scouting. The first level is similar to DSA, making a move based on π_i in Fig. 2c. As illustrated in Fig. 3d, the second level is expanding the node S_E based on the average of the f values on the longest path from S_E to a leaf after a subtree is scouted by A* with a pre-fixed number nodes expanded, that is, this average is used in place of the original f value associated with the node S_E to guide A* search on the first level.

Moreover, AlphaGoZero also gets a priori $p = p(s, a) = p(a|s)$ by deep network and uses it in Eq. (6) to make action while there is no such a priori considered in these A* based techniques.

In Table 1(B), we further extend those techniques in Table 1(A) into DSA-E, DCA-E, DBA-E, and AlphaGoZero-E, featured by the following modifications:

- (1) *DL priori policy* deep learning network $[v_h, g](s) = f_\theta(s)$ is extended into $[v_h, g, \mathbf{p}](s) = f_\theta(s)$ to add an output for a priori policy $p = p(s, a) = p(a|s)$.
- (2) *Bayesian valuation* combining such a priori policy and turning the estimates Q, f, v_h into a sort of likelihood such that a posteriori \mathbf{q} is obtained by Bayesian formulae, then making action based on this posteriori. It degenerates back to Table 1(A) when elements of \mathbf{p} are same.
- (3) *OPEN beaming* only a part of children of s are put into OPEN based on \mathbf{q} with the rest into WAIT that is a preparatory list of storing these nodes, some of which will be moved back to OPEN once OPEN becomes empty. It degenerates back to Table 1(A) when all the children of s are put into OPEN.
- (4) *DBFS_n-A* selection* a spectrum of selection strategies can be obtained by combining the one from A* and the one from MCTS by varying $n = 0$ to $n = d$, where d is the length of the path from the root to S_E hit by DBFS. As illustrated in Fig. 3f, one

end is DBFS₀-A that hits S_E by MCTS without using A*. Generally, DBFS _{n} -A* conducts DBFS search to return back for n nodes along the path from S_E back to the root. For examples, DBFS₁-A* returns back to its father node, while A* or precisely BFS selects the best among all the children (i.e. ones indicated by double circle) as S_E ; DBFS₂-A* returns back for two nodes, while BFS selects the best among all its children and grandchildren (i.e. add in two double circled nodes) as S_E , so forth. The other end is DBFS _{d} -A that returns back to the root and becomes equivalent to A* that picks the best among OPEN.

- (5) *OPEN revision* A* search will not revise the f values in OPEN and thus do not affect future expanding, while AlphaGoZero or MCTS uses back propagation to revise Q values along the path from S_E to the root of tree, which affects the search of the next circling to hit a node to expand. Also, we may integrate this idea to revise the f values in OPEN per expanding such that the next expanding will be affected. As illustrated in Fig. 3g, after expanding S_E and getting the best value f^* of children's f values, we back-propagate f^* to revise the f values of nodes one by one along its path $path_b$ back to the root of tree, e.g. the f value of s_0 is updated by a weighted average of its old value and f^* , and then the f value of s_1 is updated by a weighted average of its old value and the new f value of s_0 . Precisely, we make a revision by

$$\begin{aligned} f^{new}(s_f) &= (1 - \eta)f^{old}(s_f) + \eta f^{new}(s_s), \text{ on backward path} \\ f^{new}(s_s) &= (1 - \eta)f^{old}(s_s) + \eta f^{new}(s_f), \text{ on forward path,} \end{aligned} \quad (7)$$

where $1 > \eta > 0$. On a backward path, the notation (s_f, s_s) is moved up after each step, until the f value of the tree root is revised. Similarly, on a forward path, the notation (s_f, s_s) is moved down after each step, from each node on $path_b$ to reach a node in OPEN.

Particularly, DBA-E is an example of *past-future integrated (PFI)* mechanism that selects the node to expand by jointly using information within and outside the red coloured boundary drawn in Fig. 2b. 'Future' information outside the closure is collected via one optimal scouting path by A* search and the average of f values along this path, while 'past' information inside the closure is collected via back propagation by MCTS. Generally, there could be different detailed forms of PFI mechanism, for which further study may deserve.

Each of search techniques in Table 1(B) degenerates back to its counterpart given in Table 1(A) when all the above modifications are shut off. Whether each of these modifications gets in action is task dependent, and correspondingly we may have various special cases in place of ones in Table 1(B). They all can be regarded as examples of deep bidirectional intelligence addressed at the beginning of this paper. Deep implementation of problem solving search is driven by deep abstraction or A-type mapping via deep learning network in harmony. These exemplars constitute a family that can be shortly named deep IA-search to indicate not only its nature of deep IA harmony but also its feature of making scouting aided search.

Deep learning, path consistency, and domain knowledge embedding

Similar to that made in AlphaGoZero (Silver et al. 2017), training DL network $f_\theta(s)$ starts once search process finally reaches a terminate node. As illustrated in Fig. 2c, this search results in a PATH that consists of all the nodes selected step by step from the initial state to the terminate node. For each $s \in \text{PATH}$, the associated $g^*(s), v_h^*(s)$ are obtained from the environment, based on which we update the parameters θ to modify $\mathbf{p}(s), g(s), v_h(s)$ by gradient descending to minimise the loss function:

$$\begin{aligned} L(\theta) &= |\theta|^{\gamma_r} + \sum_{s \in \text{PATH}} [w_s L_s(\theta) + w_c L_s^c(\theta)], \\ L_s(\theta) &= -\pi(s) \ln \mathbf{p}(s) + |g(s) - g^*(s)|^{\gamma_e} + |v_h(s) - v_h^*(s)|^{\gamma_e}, \end{aligned} \quad (8)$$

with $\gamma = 2$ for the L_2 error and $\gamma = 1$ for the L_1 error, where w_s , and w_g are nonnegative weighting parameters.

Ignoring the reward in Eq. (4) or the regret in Eq. (2), we discard $|g(s) - g^*(s)|^{\gamma_e}$ and let $w_c = 0$, $L(\theta)$ returns back to the same loss function used in Ref. Silver et al. (2017). In general, the above $L(\theta)$ generalises along two directions. First, the reward or regret is considered such that it becomes applicable to problem solving tasks beyond games like Go, e.g. tasks traditionally considered by A*. Second, $L_s^c(\theta)$ is added with $w_c > 0$ to enhance a nature named as path consistency, that is, *f values on one optimal path should be identical*. Previously, OPEN revision by Eq. (7) is actually rooted from this nature. In sequel, we use this nature to improve learning.

To restrict that the values of $f(s)$ predicted by DL network should deviate as less as possible on PATH, we may minimise the following loss function:

$$L_s^c(\theta) = |f(s) - f^*(s)|^\gamma, \quad (9)$$

where $f^*(s)$ is a moving average of f values of states within a segment window $W_s \subseteq \text{PATH}$ with $s \in W_s$, while the window width may vary as the state moves, e.g. it may grow gradually from 1 at the root to some value and then shrink gradually back to 1 as the terminate node of the PATH.

Learning is performed in two phases. The main phase is updating θ to minimise the overall $L(\theta)$ after reaching a terminate node, where $v_h^*(s) = 0$ such that $f^*(s) = g^*(s)$ becomes the actual reward or regret received from the environment. There is also a complementary phase made before reaching terminate node. We let $w_s = 0$ to shut off $L_s(\theta)$ that is unavailable yet because the value v_h of each node on this incomplete PATH is unknown. Still, we have $f_\theta(s)$ given by DL network, together with $f^*(s)$ given by the average of such $f_\theta(s), \forall s \in W_s$, based on which we may update θ to minimise a part of $L(\theta)$ to ensure path consistency.

The above proposal is directly applicable to those tasks that seek whether or not certain conditions are satisfied, e.g. Go game, Boolean satisfiability problem, and various constraint satisfaction problems. In these cases, each action $s \rightarrow a$ does not contribute the final result incrementally, and there is no additive nature $f = g + v_h$. The situation is degenerated into $g(s) = 0$ and $f(s) = v_h(s)$. Still, path consistency holds and it follows from Eqs. (8) and (9) that learning is made by minimising

$$L(\theta) = |\theta|^\gamma + w_s \sum_{s \in \text{PATH}} [|v_h(s) - v_h^*(s)|^\gamma - \pi(s) \ln \mathbf{p}(s)], \quad (10)$$

where $v_h^*(s)$ is obtained by Eq. (9) with f replaced by v_h and w_c replaced by w_s .

Additionally, there are also task-dependent constraints. One classic example is the travelling salesman problem (TSP) that not only finds a minimum path of passing a number of cities but also satisfies the constraint of visiting each city once and only once. Another example is considering a portfolio of assets, e.g. stocks, bonds, currencies, and metals, etc. Changing holding of one or more assets leads to a new state, incurring for profit gain or loss r_t . Within a given period, the task of changing states for a best profit can be formulated as a tree search problem.

Typically, tasks of TSP and portfolio management as well as many problem solving problems are formulated in a representation that may facilitate conventional computing but is difficult for making deep learning network to distinguish different states because the representation is usually discrete, symbolic, and conceptual. One potential direction to tackle the problems is what we may call *feature enrichment*. Proceeding along an opposite direction of feature extraction or selection that compresses data, removes redundancy, and extracts important information, we enrich compressed or simplified representation by restoring topological information, neighbourhood information, and association information, etc. The enriched representation per state can be either or both of the following two formats:

A manifold in high dimensional space Each state denotes a manifold, a convex set, and a cluster of points in a high dimensional space. Similar states correspond topologically to neighbored manifolds or clusters.

A 2D or more image Each state denotes a collection of images either varying continuously or sharing common critical features. Being different from the above vector representation, image representation is good at encoding dependent structure related to element locations.

Some possible examples are suggested as follows:

- For portfolio management, we turn time series of assets into to images, with the help of time–frequency analysis, such as short-time Fourier transform (STFT).
- For TSP problem, a typical representation of a state is a contour or trace from the starting city to the end city in a 2D image with locations of n cities. We may turn the 2D contour by 3D time–frequency analysis, scale space representation, and wavelet analysis, etc.
- For natural language understanding, we associate words with its corresponding speech signals or image patterns, and then treat the signals and patterns as above to generate image inputs to deep learning networks. Also, we may embed each word or a parsing tree of a sentence into high dimensional space to generate vector input to deep learning networks.

Deep IA-infer

A process of tree search such as AlphaGo may be regarded as a process that infers the statement “the skill of player A is higher than player B” is true, which is proved or disproved at a terminal state of tree search. Though output is deterministically either of win, lost, and tie, the result has uncertainty since it comes from just one game. Uncertainty may be reduced by playing a number of games. Also, the value v is attached to each state as a belief value against uncertainty. Finally, we can backtrack the path from the tree root to the terminal state, and get to know the reason of winning.

Generally, search process of solving Boolean satisfiability and other constraint satisfaction problems may be regarded as uncertainty reasoning process of inferring a statement via checking whether certain conditions are satisfied.

During the last wave of AI studies in the eighties of the last century, tree search not only found many applications in the areas of pattern recognition (Xu et al. 1989), but also took important roles in problem solving tasks, especially in the hot topic called expert system. Starting at a set of preconditions towards a set of ending conditions that specify a consequence, the reasoning process is actually a tree search, with each elementary unit being a star structure. Each state associates with a number of attributes that specify an antecedent, and several production rules that match this antecedent act as branches emitted from this state. After reaching the state that represents a targeted consequence, a path backtracked to the tree root will provide a sequence of IF-THEN rules that explain the reasoning process.

Regrettably, studies on these tasks gradually faded out since the late eighties in the last century because implementation of tree search confronted great challenges in computing complexity. Nowadays, AlphaGoZero and each of deep IA-search techniques in Table 1(B) as well as learning by Eq. (10) arise a new direction for rebooting these early studies, with the help of deep learning and recent advances on tree search.

The tree expanding process $\{s_t, y_t, h_t\}, t = 1, \dots, T$ records a reasoning sequence, where y_t consists of attributes associated with state s_t , each transition $s_t \rightarrow s_{t+1}$ denotes an implementation of production rule with its antecedent best matching y_t among several production rules and with its postcedent attached to s_{t+1} as its attributes, and $h_{t-1} \rightarrow h_t$ represents the corresponding change of uncertainty that the search is guaranteed to reach a targeted consequence. The matching between y_t and the antecedent of each rule is usually inexact and will increase uncertainty. There is a sequence of actions a_1, a_2, \dots, a_t that controls tree expanding process in order to maintain low uncertainty. In such a way, reasoning is implemented with the help of techniques of deep IA-search, which is here referred to as Deep IA-Infer.

The star structure illustrated either in Fig. 4b or by the shadowed area in Fig. 4a takes a fundamental role in tree search or tree reasoning. Typically, each star structure (i.e. a number of edges emitted from the state) is pre-specified according to domain-dependent knowledge, and also possible to be identified from data in some cases. Then, certain quantities, such as $p_\rho(a|s)$, $Q(s, a)$, and $p(s_{t+1}|s_t)$, are learned from data collected either in advance or during search.

In general, there are two types of star structure. As illustrated by the shadowed area in Fig. 4a, each link of the first type acts as a ‘perceptor’ or ‘controller’ that perceives and maps the current state information into one of actions. Such actions can be regarded as

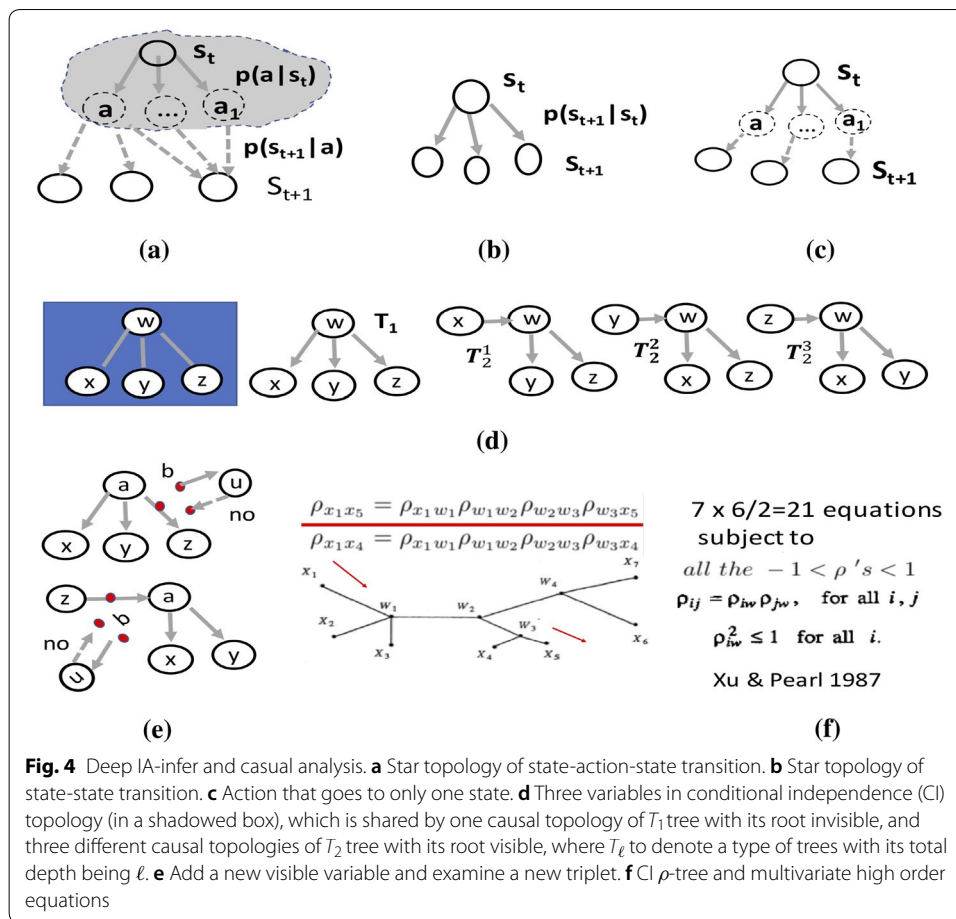


Fig. 4 Deep IA-infer and casual analysis. **a** Star topology of state-action-state transition. **b** Star topology of state-state transition. **c** Action that goes to only one state. **d** Three variables in conditional independence (CI) topology (in a shadowed box), which is shared by one causal topology of T_1 tree with its root invisible, and three different causal topologies of T_2 tree with its root visible, where T_ℓ to denote a type of trees with its total depth being ℓ . **e** Add a new visible variable and examine a new triplet. **f** CI ρ -tree and multivariate high order equations

an example of abstract representation such as decision index, features, and control signal, and thus may be named as A-type link. Each link of the second type acts as an ‘actuator’ that is driven by the selected action to implement certain mechanism that moves the state s_t to the next s_{t+1} , and thus may be named as I-type link.

Both types usually contain randomness and uncertainty and, thus, are modelled by probabilities $p(a|s_t)$ and $p(s_{t+1}|a)$, respectively. Collapsing A-type links and I-type links between a state to its direct descendant states, the situation illustrated in Fig. 4a will be simplified to the star structure illustrated in Fig. 4b. However, such a collapsing degrades the representation ability except the special case shown in Fig. 4c, such as ones considered by AlphaGoZero and deep IA-search.

Casual analysis: model-based versus data-based approaches

When we say that changing x causes y changing, we observe the natures as follows:

Dependence we observe dependence between changes of x and y .

Directional x changes before y changes for at least an infinitesimal interval.

De-interference changes of y are caused by neither itself or its environment.

Also, either or both of x and y is not limited to be a scalar, but possibly a vector or a set of variables.

The above three 3-D natures jointly tell that the changes of y are indeed caused by x at least partially, in a general scope but a weak sense. We may observe a stronger causality in a more restricted scope that y changes with x in a regular manner, e.g. satisfying certain dynamic law or following a particular mechanism, which actually leads to one current popular direction of casual analysis on nonexperimental data, featured with *model-based approach*. That is, we attempt to fit nonexperimental data by a model that generates y from x in a manner usually regarded as causal, e.g. possessing at least the above three natures, and interpreting according to the model that there is a causality underlying the data observed. Typically, the topology of such a model is a Directed Acyclic Graph (DAG). Examples include not only Bayesian networks (Pearl 1988). Examples include not only the classical structure equation model (SEM) (Wright 1921; Pearl 2010), but also LiNGAM (Shimizu 2006), post-nonlinear (PNL) model (Zhang and Hyvärinen 2009), and additive noise model (ANM) (Hoyer et al. 2009).

This direction of studies has two critical problems. The first is how to judge whether data fit the model well. A conventional fitting error or likelihood may not be enough. A small fitting error may be obtained in a risk of undermining or violating causality. The second is how to judge whether a model describes causality well. There lacks a numeric measure yet. Instead, one typically checks whether some restrictions are satisfied. Current ways that tackle the two problems are estimating two or a few candidate models to minimise error and then picking one via checking which one satisfies restrictions well.

Yet, there is still a distance towards a best solution from the perspectives of both minimising description error and ensuring causality, for which we need to seek a causal orientated fitting error measure. Such a measure should cover not only the conventional fitting error but also deviation from causality. The latter is about deviation from the DAG topology that models causality, related to not only the complexity of DAG but also whether the casual direction is consistent to the corresponding directions of the DAG topology. Therefore, a reasonable guess about such a measure is somewhat a kind of directional generalisation error.

Actually, the model-based approaches represent just one category of a dichotomy of casual analysis. The other category is *data based approach*, discovering causality underlying data without assuming a data generative model but with efforts that ensure the 3-D natures. One example is Rubin causal model (RCM) (Rubin 1974; Rubin and John 2011). The key idea is intervening the change of designated action, e.g. typically changing between two states usually denoted by the case $x = 1$ and the control $x = 0$, and then observing whether the corresponding effect of y ensures the first two natures, while the nature of *de-interference* is considered by weighted adjustment via the conditional independence (CI) $x \perp y \mid u$ on a covariate that describes the environment u or its propensity score that represents sufficient dimension reduction.

Another example came even much early. Considering that x and y share one common factor u with $x \perp y \mid u$, it follows from Ref. Reichenbach (1956) that the causality is described by either of three casual tree topologies that share a common CI topology $x - u - y$. First, $x \leftarrow u \rightarrow y$ is a simple tree of type T_1 with its root being hidden and two leaf nodes located on the first layer, where for simplicity we use T_ℓ to denote a type of trees with its total depth being ℓ . Either of the latter two $x \rightarrow u \rightarrow y$ and $x \leftarrow u \leftarrow y$ is a simplest tree of type T_2 with its root being visible and each of two layers having only one node.

Using one CI testing, we are able to recover the CI topology but unable to determine which one among the three causal topologies, i.e. we are unable to orientate the direction of each edge. Such a CI test-based study applies to generally the cases with many variables too. The best known examples are the inductive causation (IC) algorithm (Pearl 1991) and its refined counterpart named PC algorithm (Spirtes and Glymour 1991). The latter starts with a completely connected graph and deletes recursively edges based on CI tests, resulting in the CI topology. Next, nondirectional edges of every V-structure are turned into directional edges. However, merely a part of edges can be orientated in such a way. The rest edges still remain non-directional. One way is orientating each edge a direction randomly in consistence with conditional independence.

Generally speaking, CI nature can recover CI topology but is unable to orientate every edge direction. We may use an approach similar to RCM for the direction of each of un-orientated edge. Also, techniques used in the above-mentioned LiNGAM, PNL, and ANM as well as the SADA (Cai et al. 2013) may also be adopted for this purpose.

Discovering star structure and TPC causal learning

The CI topology obtained above consists of merely visible variables, which is not enough in real situations where certain hidden factors need to be taken in consideration. One early study is made by Pearl (1986) on the problem of binary visible nodes together with a number of binary hidden variables.

One example is identifying whether $p(x_1, x_2, x_3)$ is in a star structure as follows:

$$p(x_1, x_2, x_3|w) = p(x_1|w)p(x_2|w)p(x_3|w), \quad (11)$$

as illustrated within the shadowed box in Fig. 4d. It is found that a necessary condition for identifying CI topology is simply that correlation coefficients ρ_{ji} between x_j, x_i obey the following triangle inequalities:

$$\rho_{jk} \geq \rho_{ji}\rho_{ik}, \quad \text{with } \rho_{jk}\rho_{ik}\rho_{ji} \geq 0, \forall i \neq j \neq k. \quad (12)$$

Subsequently, the study was extended to Gaussian visible nodes x_1, x_2, x_3 in Refs. Xu (1986a); Xu and Pearl (1987), in which it was found that the necessary and sufficient condition for identifying this CI topology is satisfying the above triangular inequalities.

Actually, Eq. (12) came from Eqs. (14) and (15) in Ref. Xu and Pearl (1987), i.e. the following property about pairwise correlation coefficients (or shortly ρ -parameters)

$$\rho_{ji} = \rho_{iw}\rho_{jw}, \forall i, j \quad \text{and} \quad \rho_{iw}^2 \leq 1, \forall i, \quad (13)$$

which describes the constraints on two kinds of ρ -parameters located on each side of the equality. The kind on the left hand is attached to a visible edge and directly known from observation data, while the other kind on the right hand is attached to an invisible edge and actually acts as one unknown parameter in the star structure, i.e. the causal model we consider.

Both Eqs. (12) and (13) apply to not only the triplet by Eq. (11) but also generally a star topology with more than three visible nodes as follows:

$$p(x_1, x_2, \dots, x_n|w) = p(x_1|w)p(x_2|w) \cdots p(x_n|w). \quad (14)$$

Table 2 TPC causal learning

Solving joint equations for CI ρ -tree by Theorem 2 and for CI ρ -DAG by Theorems 3 and 4			
Joint equations	Topology identification (T-phase)	Parameter reestimation (P-phase)	Causal ρ -tree search (C-phase)
No solution	Inconsistent with data	NA	NA
Unique solution	As the necessary and sufficient condition for identifying this CI topology of ρ -tree or ρ -DAG, which is an equivalence class of a number of causal ρ -trees or ρ -DAGs. At least, one of them models data well	Each in this equivalence class is modeled in SEM equations with coefficients as parameters that are re-estimated via iterating one SEM-based constrained sparse optimisation with coefficients get in T-phase as initialisation	Search the best one among the equivalence class with each one enumerated by a search strategy, estimated in P-phase, and evaluated by a measure that considers both best-fit and causality to get a directional generalisation error.
Many or infinite many solutions	As a necessary condition that this CI topology satisfies		

In a summary, we have the following theorem:

Theorem 1 *To identify the star topology by Eq. (14), the satisfaction of the triangular inequalities by Eq. (12) or equivalently the equalities by Eq. (13) is (a) the necessary condition for random variables x_1, x_2, \dots, x_n in general, and (b) the necessary and sufficient condition for Gaussian variables x_1, x_2, \dots, x_n in particular.*

Even when the theorem is satisfied, it should be noted that there are still two types of indeterminacy. First, though the satisfaction of Eq. (13) uniquely specifies the star topology, this topology is shared by an equivalence class that consists of not only one causal topology of T_1 tree with its root hidden but also three different causal topologies of T_2 tree with its root visible, as illustrated in Fig. 4d. Second, the solution for the triplet $(\rho_{12}, \rho_{13}, \rho_{32})$ to satisfy Eq. (13) is not unique but may be infinite many.

Typically, identifying star or CI topology and estimating unknown parameters are handled non-separately (Spearman 1904; Wright 1921; Anderson and Rubin 1956; Pearl 1986; Pearl 2010; Shimizu 2006; Zhang and Hyvärinen 2009; Hoyer et al. 2009). Differently, the above theorem implies a possibility of separating *topology identification and parameter estimation*. Taking the issue of equivalence class in consideration too, we get a *three phase* causal learning approach as summarised in Table 2.

The first phase identifies the CI topology, or shortly *T-phase*. Simply, a star topology may be identified in two ways. One is model free based on Eq. (12), involving merely estimating ρ_{ij} 's from samples. However, for a finite number of samples, the resulted ρ_{ij} 's suffer from randomness and inaccuracy. Certain results about distribution function of correlation coefficient are available in the existing literature, based on which we may further consider Eq. (12) probabilistically as follows:

$$P(\rho_{jk} \geq \rho_{ji}\rho_{ik}) > 1 - e, \quad \forall i \neq j \neq k. \quad (15)$$

with a small $e > 0$ as a tolerance threshold.

The other way is based on a start causal model with a set ρ of n unknown parameters ρ_{iw} 's, from which we obtain Eq. (13) that consists of $0.5n(n-1)$ joint equations.

As summarised in Table 2, we examine the possible outcomes of the joint equations to identify the start causality. The best outcome is that the joint equations have a unique solution, which identifies that the casual tree topology in consideration is among an equivalence class of four simple causal ρ -trees as illustrated in Fig. 4d, but fails to identify uniquely which one.

Alternatively, we may let $\xi_{ij} = -\ln |\rho_{ji}|$ and turn Eq. (13) into

$$\xi_{ij} = \xi_{iw} + \xi_{jw}, \forall i, j \quad \text{and} \quad \xi_{iw} \geq 0, \forall i, \quad (16)$$

which consists of $0.5n(n-1)$ joint linear equations of n nonnegative unknown variables. After getting an estimate of ξ_{iw} , we may get $|\rho_{iw}| = \exp(-\xi_{iw})$ and then recover the signs from Eq. (13).

The second phase is named *P-phase* that estimates ρ of all the unknown ρ -parameters by

$$p(x_1, x_2, \dots, x_n) = \int p(x_1, x_2, \dots, x_n, w) dw, \quad (17)$$

for a given casual tree topology taken from the equivalence class. Specifically, we may use SEM equations to model the casual tree topology, with coefficients as parameters that are re-estimated via iterating a constrained sparse optimisation with coefficients get in T-phase as initialisation.

When the joint equations from this casual tree topology get many or infinite many solutions in T-phase, it only partially supports that this casual tree topology is among the equivalence class. Still, we may approximately perform P-phase to model this casual tree topology. To reduce this indeterminacy and also inaccuracy caused by a finite number of samples, certain structural constraint and sparse regularisation can be imposed.

The third phase is named *C-phase* that aims at choosing the best one among the equivalence class, for which we need a search strategy to enumerate among the equivalence class and also a measure to evaluate both best-fit and causality to get a directional generalisation error.

In a summary, the separation principle that was firstly revealed in Refs. Xu (1986a); Xu and Pearl (1987) has been here further developed into a method of making three phase learning, namely T-phase, P-phase, and C-phase, or shortly TPC causal learning.

Structuring causal trees: TRIPLET, STAR, and their recursions

We address further implementations by starting with the triplet of three visible nodes x_1, x_2, x_3 . Without losing generality, we normalise each x_i to be zero mean and unit variance, and also assume that w has zero mean and unit variance. From Theorem 1(c) or Eqs. (9) and (10) in Ref. Xu and Pearl (1987), it follows that Eq. (13) is associated with the following linear regression or the structure equation model (SEM):

$$\begin{aligned} \text{For } T_1 : [x_1, x_2, x_3]^T &= [\rho_{1w}, \rho_{2w}, \rho_{3w}]^T w + [e_{1w}, e_{2w}, e_{3w}]^T, \quad Ee_{iw}w = 0, \forall i, \\ \text{For } T_2^1 : [x_2, x_3]^T &= [\rho_{2w}, \rho_{3w}]^T \rho_{1w}x_1 + [\rho_{2w}e_{w1} + e_{2w}, \rho_{3w}e_{w1} + e_{3w}]^T, \\ \text{For } T_2^2 : [x_1, x_3]^T &= [\rho_{1w}, \rho_{3w}]^T \rho_{2w}x_2 + [\rho_{1w}e_{w2} + e_{1w}, \rho_{3w}e_{w2} + e_{3w}]^T, \\ \text{For } T_2^3 : [x_1, x_2]^T &= [\rho_{1w}, \rho_{2w}]^T \rho_{3w}x_3 + [\rho_{1w}e_{w3} + e_{1w}, \rho_{2w}e_{w3} + e_{2w}]^T, \end{aligned} \quad (18)$$

where the 1st line equation is straightforward to understand. In order to understand the last three lines, we need to add one equation $w = \rho_{iw}x_i + e_{wi}$, with its variance of e_{wi} being $1 - \rho_{iw}^2$. Example the last line equation comes from $x_1 = \rho_{w1}w + e_{1w}$, $x_2 = \rho_{2w}w + e_{2w}$ with $w = \rho_{3w}x_3 + e_{w3}$, which produces $x_1 = \rho_{1w}\rho_{3w}x_3 + \rho_{w1}e_{w3} + e_{1w}$ and $x_2 = \rho_{2w}\rho_{3w}x_3 + \rho_{2w}e_{w3} + e_{2w}$. The variance of $\rho_{w1}e_{w3} + e_{1w}$ is $\rho_{w1}^2(1 - \rho_{w3}^2) + 1 - \rho_{w1}^2 = 1 - \rho_{w1}^2\rho_{w3}^2$ and $\rho_{w2}e_{w3} + e_{2w}$ is $1 - \rho_{w2}^2\rho_{w3}^2$.

In the general case that $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ and thus $\mathbf{e}^w = [e_{1w}, e_{2w}, \dots, e_{nw}]^T$, we have one causal topology of T_1 tree with its root being the hidden variable w , and n different causal topologies of $T_2^i, i = 1, \dots, n$ trees with its root being the visible variable x_i , respectively. We may summarise the equations in Eq. (18) into

$$\begin{aligned} \text{For } T_1 : \mathbf{x} &= \boldsymbol{\rho}_w w + \mathbf{e}^w, Ew = 0, E\mathbf{e}_w w = 0, E\mathbf{e}_w = 0, \\ \boldsymbol{\rho}^w &= [\rho_{1w}, \dots, \rho_{nw}]^T, E\mathbf{e}_w \mathbf{e}_w^T = \text{diag}[1 - \rho_{\ell w}^2], \\ \text{For } T_2^i : \mathbf{x}_{-i} &= \boldsymbol{\rho}_{-i}^w \rho_{iw} x_i + \boldsymbol{\rho}_{-i}^w e_{wi} + \mathbf{e}_{-i}^w, \mathbf{x}_{-i} = \mathbf{x} - \{x_i\}, \\ E[\boldsymbol{\rho}_{-i}^w e_{wi} + \mathbf{e}_{-i}^w][\boldsymbol{\rho}_{-i}^w e_{wi} + \mathbf{e}_{-i}^w]^T &= \text{diag}[1 - \rho_{\ell w}^2 \rho_{iw}^2]. \end{aligned} \quad (19)$$

where $\text{diag}[d_\ell]$ denotes a diagonal matrix with d_ℓ being its ℓ -th element. Moreover, \mathbf{x}_{-i} comes from taking x_i out of \mathbf{x} and correspondingly we have $\mathbf{e}_{-i}^w, \boldsymbol{\rho}_{-i}^w$. Also, there is an additional equation $w = \rho_{iw}x_i + e_{wi}$ for the cases of T_2^i .

Given a set of samples of \mathbf{x} , we may use the weighted least square error method on either of the equations for T_1 and the equations for T_2^i to get a lost function $-\mathcal{L}(\boldsymbol{\rho}, T)$ for measuring how well the causal tree topology T and the corresponding SEM equations fit samples. Moreover, let all e_{iw} 's and e_{wj} 's as well as w to be Gaussian with zero means and mutually uncorrelated, we may estimate distribution $p(\mathbf{x})$ and $p(\mathbf{x}_{-i}|x_i)$ and get a likelihood function $\mathcal{L}(\boldsymbol{\rho}, T)$ to measure how well the model T fits. Then, $\boldsymbol{\rho}$ is estimated by the following constrained optimisation:

$$\max_{\boldsymbol{\rho}, T} \{L(\boldsymbol{\rho}, T) - \gamma \sum_i |\rho_{iw}|^q\}, T \in \{T_1, T_2^i, i = 1, \dots, n\} \text{ subject to Eq. (13)}. \quad (20)$$

We choose the best $\boldsymbol{\rho}^*$ in the best model T^* among all the candidates. The above second term adds in regularisation with a strength $\gamma > 0$, with $q = 1$ or 2 for L_1 and L_2 regularisation, respectively. This regularisation prefers simplicity, i.e. ρ_{iw}^2 or $|\rho_{iw}|$ is as smaller as possible. Interestingly, this simplicity is equivalent to independence because $\rho_{iw} = 0$ either implies independence of Gaussian variables x_i, w or acts as a necessary condition for independence of variables x_i, w in general.

In sequel, we sketch a TRIPLET procedure for possible real applications:

Step 1 Select a set V of significant variables, where each $x \in V$ is selected from the set U of all the variables if x is regarded as “significant” by either a hypothesis testing or according to a criterion. One example is that each $x \in V$ is a significant SNV selected from U that consists of all the SNVs in a GWAS study. The other example is that each $x \in V$ is a significant biomarkers selected from U that consists of expressions of all the genes in genomics analyses.

Step 2 For every $x \in V$, we screen every triplet (x, y, z) by enumerating every pair (y, z) from the difference set $U - V$. This triplet is ignored if it is already in

the set T^d that accommodates all the detected triplets, otherwise, u is examined by either one inequality test based on Eqs. (12) and (15) or one equality test based on Eqs. (13) and (16), and then it is added into T^d if the test is passed.

Step 3 For each triplet $(x, y, z) \in T^d$, we solve Eq. (20) and choose the best ρ^* in the best model among the four candidates.

This TRIPLET procedure finds a set of triplets and puts in T^d .

We further extend a known triplet $a - xyz$ into star structure with more than three edges. Putting a triplet (x, y, z) into S , we implement a STAR procedure as follows:

- Pick away from $U - S$ a visible variable u that has the strongest correlation with either one of variables in S , and then perform T-phase to examine a star that consists of u and all the variables in S by either an inequality test based on Eqs. (12) and (15) or an equality test based on Eqs. (13) and (16).
- If the test fails, discard u . If success, put u into S and choose candidate causal trees by choosing $a \rightarrow u$ or $u \rightarrow a$ to avoid a V-structure.
- Make P-phase to solve Eq. (20) and choose the best ρ^* in the best model among the candidates obtained above.
- Examine whether a pre-specified terminating condition is satisfied, if yes, stop; otherwise, goto the beginning line above.

One terminating condition can be simply that the number of elements in S becomes larger than a threshold. The other is that every variable in $U - S$ gets a weak correlation with every variable in S .

Moreover, we may further extend a triplet topology into a general causal tree structure by recursively performing the following RECURSIVE TRIPLET:

Step 1 As illustrated in Fig. 4e, we choose one edge l_{az} on a known triplet $a-xyz$ and consider to locate at the middle of l_{az} an additional edge l_{bu} that has one end being a new hidden variable b and the other end being a new visible variable u . We thus get a new triplet $b-zua$.

Step 2 Make T-phase to test this new triplet $b-zua$ in the same way as Step 2 in the above TRIPLET procedure. If failed, l_{bu} is discarded.

Step 3 If succeed, perform P-phase by Eq. (20) on the triplet $b-zua$ and choose the best ρ^* in the best model T^* among the four candidates.

Next, we choose another new edge and repeat the same procedure, \dots , so on so forth, until all the visible nodes of not only the original triplet $a - xyz$ but also all the newly added visible nodes have been examined.

Specifically, there are different choices to pick which visible variable as u and pick which edge as l_{az} . For the former one, we may pick a visible variable that has the strongest correlation with the visible end (i.e. z) of edge l_{az} . For the latter one, we may assign each edge in the current tree an ordering index (e.g. it can be the time that the

edge is added, and the oldest is picked first). Alternatively, we may also modify Step 3 into the following two steps:

- Step 3(a)* if succeed, solve Eq. (20) on the triplet b-zua and choose the best ρ^* in the best model among the four candidates. Also, record the corresponding best fitting measure L_{az}^*
- Step 3(b)* repeat from Step 1 to Step 3 on the other two edges l_{ay} and l_{ax} , respectively. Choose the best ξ^* according to $L_{a\xi}^*$ for $\xi = x, y, z$ and finally add the edge l_{bu} to the middle of the corresponding edge $l_{a\xi^*}$.

RECURSIVE TRIPLET and the above revision suffer a problem that results in a tree with hidden nodes increasing proportionally with the number of visible nodes. One remedy is letting the visible node u of the edge l_{bu} to be one of three visible nodes in a detected triplet. In such a way, two triplets are joined if we get at least one successful test on the new triplet b-zua in Step 3. Also, it is not difficult to observe that such a joining procedure will not violate the tests already made on two previous detected triplets. Next, we choose another triplet and repeat the same procedure, ..., so on so forth, until all the detected triplets have been all examined.

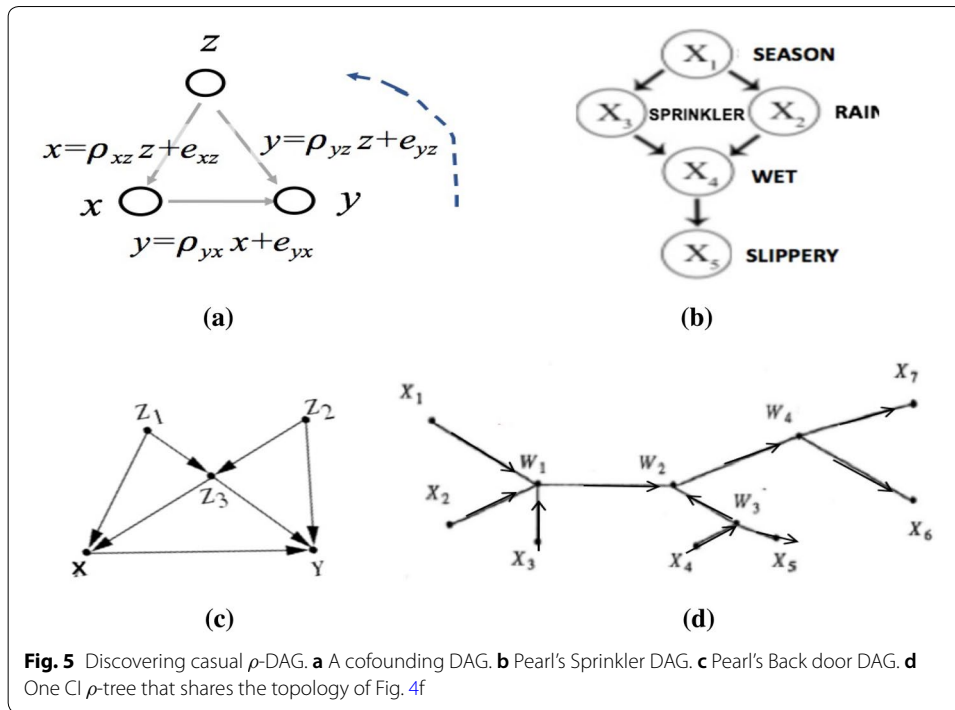
Alternatively, we may also locate the hidden centre of a triplet at the middle point of edge l_{az} . In such a way, the middle point actually forms a star topology of five edges. Even more, we may locate the hidden centre of a star with more edges at the middle point or merge the hidden centres of two triplets or stars to form a bigger star. Then, we can check this star by either an inequality test based on Eqs. (12) and (15) or an equality test based on Eqs. (13) and (16). If the test is passed, we solve Eq. (20) and choose the best ρ^* in the best model among all the candidates. So on and so forth, ..., we add a visible node, a triplet, and a star into the current tree until all the visible nodes have been examined.

CI ρ -diagram and casual ρ -tree discovery

In some real applications, a possible topology of a causal tree may come from domain knowledge. Also, we may be asked to compare causal topologies resulted from using different existing methods for causal analyses. Thus, it is also demanded to further examine whether a given topology is a CI diagram that defines an equivalence class of a number of causal trees. Beyond tree topology, a CI ρ -diagram is a Directed Acyclic Graph (DAG) or Bayesian networks, on which extensive studies have been made (Pearl 1988, 2010; Spirtes and Glymour 1993, Spirtes et al. 2000).

Illustrated in Fig. 4f, we consider a diagram with visible nodes x_1, x_2, \dots, x_n and hidden nodes w_1, \dots, w_m , in which again not only each x_i is normalised to be zero mean and unit variance but also each w_j is assumed to be zero mean and unit variance too. In this diagram, each edge is associated with the correlation coefficient ρ between its two nodes. Such a diagram is completely defined by a set of pairwise correlation coefficient ρ , shortly called ρ -diagram.

Since independence implies decorrelation (i.e. an edge can be removed if its associated correlation coefficient $\rho = 0$), a CI diagram must be a CI ρ -diagram. On the other hand, two variables x_1, x_2 may not be independent even there is no correlation between x_1, x_2 , because there may still be higher order dependence between them, i.e. a conditional decorrelation



(CD) ρ -diagram may not be a CI diagram. For simplicity, we still use the name CI ρ -diagram in a narrow sense that the diagram is a CI-diagram while we consider each edge merely by its associated correlation coefficient ρ . That is, we consider a restricted form of CI-diagram, namely CI ρ -diagram.

In sequel, we start at a special case that the diagram is a tree, namely CI ρ -tree, on which we may extend Eq. (13) into the following theorem:

Theorem 2 *Given a conditional independence (CI) undirected ρ -tree topology, considering a pair of nodes ξ, η by a undirected path $\xi - x_1 - \dots - x_m - \eta$ and adding an additional edge between ξ, η associated with $\rho_{\xi\eta}$ to form in a loop, we have*

$$\rho_{\xi\eta} = \rho_{\xi x_1} \rho_{x_1 x_2} \dots \rho_{x_{m-1} x_m} \rho_{x_m \eta}.$$

Based on this theorem, we obtain high order equations by considering every pair of visible nodes. For the example in Fig. 4f with $n = 7$ visible nodes and $k = 10$ edges, there are 21 equations with 10 unknown variables as follows:

$$\begin{aligned} \rho_{x_1 x_2} &= \rho_{x_1 w_1} \rho_{x_2 w_1} \\ \rho_{x_1 x_5} &= \rho_{x_1 w_1} \rho_{w_1 w_2} \rho_{w_2 w_3} \rho_{w_3 x_5} \\ \rho_{x_1 x_4} &= \rho_{x_1 w_1} \rho_{w_1 w_2} \rho_{w_2 w_3} \rho_{w_3 x_4} \\ \rho_{x_1 x_7} &= \rho_{x_1 w_1} \rho_{w_1 w_2} \rho_{w_2 w_4} \rho_{w_4 x_7} \\ &\dots \dots \dots \\ 7 \times 6 \times 0.5 \text{ equations} \\ \text{subject to: all the } -1 < \rho' \text{'s} < 1. \end{aligned} \tag{21}$$

As a result, examining the satisfaction of Eq. (13) is turned into examining whether this Eq. (21) is solvable, for which what summarised in Table 2 still applies. Again, getting a

unique solution in T-phase identifies that the CI undirected ρ -tree in consideration is indeed among an equivalence class.

Let $\xi = -\ln |\rho|$, it follows from Eq. (21) that we further have

$$\begin{aligned} \xi_{x_1 x_2} &= \xi_{x_1 w_1} + \xi_{x_2 w_1} \\ \xi_{x_1 x_5} &= \xi_{x_1 w_1} + \xi_{w_1 w_2} + \xi_{w_2 w_3} + \xi_{w_3 x_5} \\ \xi_{x_1 x_4} &= \xi_{x_1 w_1} + \xi_{w_1 w_2} + \xi_{w_2 w_3} + \xi_{w_3 x_4} \\ &\dots\dots\dots \\ &\text{subject to : all the } \xi' \text{'s} > 0. \end{aligned} \quad (22)$$

We recover their signs of ρ 's by putting them into Eq. (21). Therefore, examining the satisfaction of Eq. (13) can also be turned into examining whether Eq. (22) is solvable and the signs of its solutions are recovered.

Also, Eq. (22) can be further turned into solving the following linear programming:

$$\begin{aligned} \max \sum &\text{all the } \xi' \text{'s, subject to} \\ \xi_{x_1 x_2} &= \xi_{x_1 w_1} + \xi_{x_2 w_1} \\ \xi_{x_1 x_5} &= \xi_{x_1 w_1} + \xi_{w_1 w_2} + \xi_{w_2 w_3} + \xi_{w_3 x_5} \\ \xi_{x_1 x_4} &= \xi_{x_1 w_1} + \xi_{w_1 w_2} + \xi_{w_2 w_3} + \xi_{w_3 x_4} \\ &\dots\dots\dots \end{aligned} \quad (23)$$

Given a undirected CI ρ -tree topology identified, we may further seek directed casual ρ -tree topologies, e.g. as illustrated in Fig. 5. One road is using IC algorithm (Pearl 1991) and PC algorithm (Spirtes and Glymour 1991). The other road is directly identifying whether a given directed causal ρ -tree with help of the following Theorem 3.

As illustrated in Fig. 5a–c, a pair of nodes ξ, η is said to be directionally correlated if the pair is linked by a path in pattern $\leftarrow \leftarrow \dots \leftarrow_j \rightarrow \rightarrow \dots \rightarrow$, where j can locate at ξ or η as well as any middle point.

Theorem 3 *Given a directed ρ -tree topology, considering a pair of nodes ξ, η that are directionally correlated by a path $\xi \leftarrow x_1 \leftarrow \dots \leftarrow x_j \rightarrow \dots \rightarrow x_m \rightarrow \eta$, we have $\rho_{\xi\eta} = \rho_{\xi x_1} \rho_{x_1 x_2} \dots \rho_{x_{j-1} x_j} \rho_{x_j x_{j+1}} \dots \rho_{x_{m-1} x_m} \rho_{x_m \eta}$.*

Based on this theorem, we can perform T-phase to check whether a given directed ρ -tree topology is consistent with a given set of samples from visible nodes. Following the procedure similar to turning causal star topologies into Eqs. (18) and (19), we turn each directed casual ρ -tree into its corresponding SEM equations, based on which we perform P-phase to solve the best ρ^* for performing the optimisation by Eq. (20). Based on these SEM equations, we may infer the first-order statistics of variables, subject to errors propagated along the paths within tree, e.g. the previous $\rho_{2w} e_{w1} + e_{2w}$, $\rho_{3w} e_{w1} + e_{3w}$ for the cases in Fig. 4d.

The information flows in a directed ρ -tree topology as illustrated in Fig. 5b are diverging from tree's root or hidden nodes to the visible nodes, thus, it may be called *diverging causal tree*. Obviously, it is a generative model or Ying causal model, implementing an I-type mapping. Revising the direction of every edge, the topology

becomes one that may be named *converging causal tree*, as illustrated in Fig. 5a, in which the information flows are converging from the visible nodes to the tree's root. Also, it is called representative model or Yang causal model, implementing an A-type mapping. Strictly speaking, such a converging causal tree is a Directed Acyclic Graph (DAG) but no longer a directed tree, for which Theorem 3 does not apply.

In comparison with structuring causal tree incrementally by TRIPLET, STAR, and their recursions, the above casual ρ -tree approach balances estimations of parameters ρ in a systematic manner. Yet, there lacks an effective technique for performing C-phase and especially for enumerating all the candidate causal ρ -tree topologies. Still, it is useful to practical needs of not only examining one or more topologies obtained from domain knowledge but also comparing causal topologies resulted from different existing methods.

Casual ρ -DAG discovery: Yule–Simpson Paradox, SPRINKLER, and BACK-DOOR

We proceed from casual ρ -tree to a general formulation of causal analysis on a CI ρ -diagram or CI ρ -DAG, with each directed edge featured by a linear SEM equation. We start with one simplest case as illustrated in Fig. 5a, which is also the simple causal topology of the well-known Yule–Simpson Paradox (Pearl 2010), described by the following SEM equations:

$$x = \rho_{zx}z + e_{zx}, y = y_z + y_x, y_z = \rho_{yz}z + e_{yz}, y_x = \rho_{yx}x + e_{yx}. \quad (24)$$

Being different from a tree, there are at least one node v with its indegree $\deg^-(v)$ being bigger than 2, e.g. $\deg^-(y) = 2$. The output of such a node is the summation of the inputs $v_{u_i}, i = 1, \dots, \deg^-(v)$ along all the edges $e_{vu_i}, i = 1, \dots, \deg^-(v)$ that enters this node, that is, we have

$$v = \sum_{i=1}^{\deg^-(v)} v_{u_i}. \quad (25)$$

In Fig. 5a, we simply have $y = y_z + y_x$.

Without losing generality, we again normalise variables to zero means and unit variances. It follows from Eq. (24) that $Ezy = Ez(y_z + y_x) = \rho_{yz}Ez^2 + \rho_{yx}\rho_{zx}Ez^2 = \rho_{yz} + \rho_{yx}\rho_{zx}$. In other words, the simple product in Theorem 2 is here extended into a summation of products, or shortly the PROD format is extended into a SUM-PROD format.

Treating Exy, Ezx in a way similar to Ezy , we have

$$\rho_{zy}^o = Ezy = \rho_{yz} + \rho_{yx}\rho_{zx}, \rho_{xy}^o = Exy = \rho_{yx} + \rho_{yz}\rho_{zx}, \rho_{xz}^o = \rho_{zx}. \quad (26)$$

It should be noted that the notation in the left hand explicitly uses a superscript 'o' to denote the correlation coefficient $\rho_{\xi\eta}^o = E\xi\eta$ between two observable variables ξ, η . We have $\rho_{\xi\eta}^o = E\xi\eta = \rho_{\xi\eta}$ if $\deg^-(\xi) = 1, \deg^-(\eta) = 1$, and thus both notations are interchangeable, as did previously in this paper. Such an interchangeability no longer holds here, a notation with a superscript 'o' denotes a correlation coefficient computed from two observable variables, while a notation without such a superscript denotes a parameter in the causal model we consider.

In Eq. (26), we may obtain $\rho_{zy}^o, \rho_{xy}^o, \rho_{xz}^o$ from experimental observation data, and then examine whether there is a confounding causal topology illustrated in Fig. 5a by examining the following corrected counterparts:

$$\rho_{yz} = \frac{\rho_{zy}^o - \rho_{yx}^o \rho_{zx}^o}{1 - \rho_{zx}^{o2}}, \rho_{yx} = \frac{\rho_{yx}^o - \rho_{yz}^o \rho_{zx}^o}{1 - \rho_{zx}^{o2}}, \rho_{zx} = \rho_{xz}^o. \quad (27)$$

It follows from $-1 < \rho_{yz}, \rho_{yx} < 1$ we should further check the following inequalities:

$$\begin{aligned} -1 &< \rho_{zy}^o - (\rho_{zx}^o + \rho_{yx}^o) \rho_{zx}^o \text{ and } \rho_{zy}^o + (\rho_{zx}^o - \rho_{yx}^o) \rho_{zx}^o < 1, \\ -1 &< \rho_{yx}^o - (\rho_{zx}^o + \rho_{yz}^o) \rho_{zx}^o \text{ and } \rho_{yx}^o + (\rho_{zx}^o - \rho_{yz}^o) \rho_{zx}^o < 1. \end{aligned} \quad (28)$$

which provides an interesting insight on the Yule–Simpson’s Paradox (Pearl 2010).

Case–control studies can be widely found in various practical uses, especially in computational health and bioinformatics. The purpose is observing whether changing x causes y changing, while most of practices actually observe certain association between x, y without considering whether confounding factors exist in the environment, via hypothesis test directly or indirectly based on estimating ρ_{yx}^o . The above Eqs. (27) and (28) provide a PDC method for reexamining such studies:

- Projection* pick a suspicious variable z that affects either or both of x, y . Generally, such z comes from a projection of a multidimensional vector that consists of many environmental variables, e.g. from a principal component;
- Detection* make a hypothesis test to detect the inequalities by Eq. (28) or its probabilistic version in a way similar to Eq. (15) discussed previously; if not, discard z ; if yes, goto the next step;
- Correction* put the corrected $\rho_{yz}, \rho_{yx}, \rho_{zx}$ by Eq. (27) into the original ρ value-based test to check the corrected effects $x \rightarrow y, z \rightarrow y$ and $z \rightarrow x$.

Next, we consider Pearl’s SPRINKLER DAG illustrated in Fig. 5b, where the node with $\deg^-(X_4) = 2$ is on the middle of a path $X_1 \rightarrow X_2 \rightarrow X_4 \rightarrow X_5$ and $X_1 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5$. Each $\rho_{\xi\eta}^o$ for a path $\xi \rightarrow \dots X_4 \rightarrow \dots \eta$ with $\xi \in \{X_1, X_2, X_3\}, \eta \in \{X_4, X_5\}$ involves the binary branching that enters X_4 and thus is in a summation of two products, resulting in 6 equations. There are also some paths $\xi \rightarrow \eta$ without involving X_4 . All together, we have the following equations:

$$\begin{aligned} \rho_{x_1x_5}^o &= (\rho_{x_1x_2}\rho_{x_2x_4} + \rho_{x_1x_3}\rho_{x_3x_4})\rho_{x_4x_5}, \\ \rho_{x_2x_5}^o &= (\rho_{x_2x_4} + \rho_{x_1x_3}\rho_{x_3x_4})\rho_{x_4x_5}, \rho_{x_3x_5}^o = (\rho_{x_3x_4} + \rho_{x_1x_2}\rho_{x_2x_4})\rho_{x_4x_5}, \\ \rho_{x_1x_4}^o &= \rho_{x_1x_2}\rho_{x_2x_4} + \rho_{x_1x_3}\rho_{x_3x_4}, \\ \rho_{x_2x_4}^o &= \rho_{x_2x_4} + \rho_{x_1x_3}\rho_{x_3x_4}, \rho_{x_3x_4}^o = \rho_{x_3x_4} + \rho_{x_1x_2}\rho_{x_2x_4}, \\ \rho_{x_2x_3}^o &= \rho_{x_1x_2} + \rho_{x_1x_3}, \rho_{x_2x_1}^o = \rho_{x_1x_2}, \rho_{x_3x_1}^o = \rho_{x_1x_3}, \rho_{x_4x_5}^o = \rho_{x_4x_5}, \\ &\text{subject to : all the } -1 < \rho's < 1. \end{aligned} \quad (29)$$

All the correlation coefficients $\{\rho_{\xi\eta}^o\}$ are estimated from a set of samples, and the problem of identifying whether SPRINKLER DAG is underlying these samples becomes examining whether the joint equations are solvable, according to Table 2.

There are 10 equations with 5 unknowns, that is, the problem may be over-constrained, especially when $\{\rho_{\xi\eta}^o\}$ are estimated from a set of samples. Therefore, it is better to consider one least error approach, e.g. like Eq. (23).

Assuming that the equations by Eq. (29) are consistent, we may use a part of Eq. (29) to get the following solution:

$$\begin{aligned} \rho_{x_2x_1} &= \rho_{x_1x_2}^o, \rho_{x_3x_1} = \rho_{x_1x_3}^o, \rho_{x_4x_5} = \rho_{x_4x_5}^o, \\ \rho_{x_2x_4} &= \frac{\rho_{x_2x_4}^o - \rho_{x_1x_4}^o}{1 - \rho_{x_1x_2}^o}, \rho_{x_3x_4} = \frac{\rho_{x_3x_4}^o - \rho_{x_1x_4}^o}{1 - \rho_{x_1x_3}^o}, \end{aligned} \quad (30)$$

from which we may get a necessary condition for identifying whether SPRINKLER DAG is underlying a given set of samples from the following inequalities:

$$\begin{aligned} -1 &\leq -\rho_{x_1x_2}^o + \rho_{x_2x_4}^o - \rho_{x_1x_4}^o \text{ and } \rho_{x_1x_2}^o + \rho_{x_2x_4}^o - \rho_{x_1x_4}^o \leq 1, \\ -1 &\leq -\rho_{x_1x_3}^o + \rho_{x_3x_4}^o - \rho_{x_1x_4}^o \text{ and } \rho_{x_1x_3}^o + \rho_{x_3x_4}^o - \rho_{x_1x_4}^o \leq 1, \end{aligned} \quad (31)$$

or alternatively a probabilistic version similar to Eq. (15). Additional conditions may be added by considering the satisfaction of equalities that are obtained from Eqs. (29) and (30) as follows:

$$\begin{aligned} \rho_{x_1x_5}^o &= \rho_{x_1x_4}^o \rho_{x_4x_5}^o, \rho_{x_2x_5}^o = \rho_{x_2x_4}^o \rho_{x_4x_5}^o, \rho_{x_3x_5}^o = \rho_{x_3x_4}^o \rho_{x_4x_5}^o. \\ \rho_{x_2x_3}^o &= \rho_{x_1x_2}^o + \rho_{x_1x_3}^o, \rho_{x_1x_4}^o = \rho_{x_1x_2}^o A + \rho_{x_1x_3}^o B, \rho_{x_2x_4}^o = A + \rho_{x_1x_3}^o B, \rho_{x_3x_4}^o = B + \rho_{x_1x_2}^o A, \\ A &= \frac{\rho_{x_2x_4}^o - \rho_{x_1x_4}^o}{1 - \rho_{x_1x_2}^o}, B = \frac{\rho_{x_3x_4}^o - \rho_{x_1x_4}^o}{1 - \rho_{x_1x_3}^o}, \\ &\text{subject to : all the } -1 < \rho^o \text{'s} < 1. \end{aligned} \quad (32)$$

The first line is similar to Eq. (13) for a star structure while a star is indeed a part of SPRINKLER DAG in Fig. 5b, and the other four lines impose more restrictions that require $\{\rho_{\xi\eta}^o\}$ to satisfy.

Moreover, we proceed to even complicated cases that a path from a node ξ to a node η contains some nodes with indegrees bigger than 1. For each of such nodes, all the edges entering this node need to be considered, and some of these edges may come from a path on which there is again at least one node with indegrees bigger than 1. In such cases, $\rho_{\xi\eta}^o$ actually involves a tree with multiple branches and thus is computed from a summation of multiple products. One example is Pearl's BACK-DOOR DAG illustrated in Fig. 5c. On the path $Z_1 \rightarrow Z_3 \rightarrow Y$ with $\deg^-(Y) = 3$, one edge comes X with $\deg^-(X) = 2$, the other comes from Z_3 with $\deg^-(Z_3) = 2$ too, and thus $\rho_{Z_1Y}^o$ is a summation of three products. From $\xi \in \{Z_1, Z_2\}$ to $\eta \in \{X, Y\}$, there are 4 such paths, resulting in 4 equations. Together with considering $Z_1 \rightarrow Z_3$ and $Z_2 \rightarrow Z_3$, as well as $Z_3 \rightarrow X, Z_3 \rightarrow Y$, we can get the following equations:

$$\begin{aligned}
 \rho_{z_2x}^o &= \rho_{z_3z_2}\rho_{xz_3}, \rho_{z_1x}^o = \rho_{xz_1} + \rho_{z_3z_1}\rho_{xz_3}, \rho_{z_3y}^o = \rho_{yz_3} + \rho_{xz_3}\rho_{xy}, \\
 \rho_{z_1y}^o &= \rho_{z_1z_3}\rho_{yz_3} + \rho_{xz_1}\rho_{xy}, \rho_{z_2y}^o = \rho_{z_2y} + \rho_{z_3z_2}\rho_{yz_3} + \rho_{xz_2}\rho_{xy}, \\
 \rho_{xy}^o &= \rho_{xy} + \rho_{xz_3}\rho_{yz_3} + \rho_{yz_2}\rho_{z_3z_2}\rho_{z_3x}, \\
 \rho_{z_3x}^o &= \rho_{xz_3} + \rho_{z_3z_1}\rho_{xz_1}, \rho_{z_1z_3}^o = \rho_{z_1z_3}, \rho_{z_2z_3}^o = \rho_{z_2z_3}, \\
 &\text{subject to : all the } -1 < \rho' s < 1.
 \end{aligned} \tag{33}$$

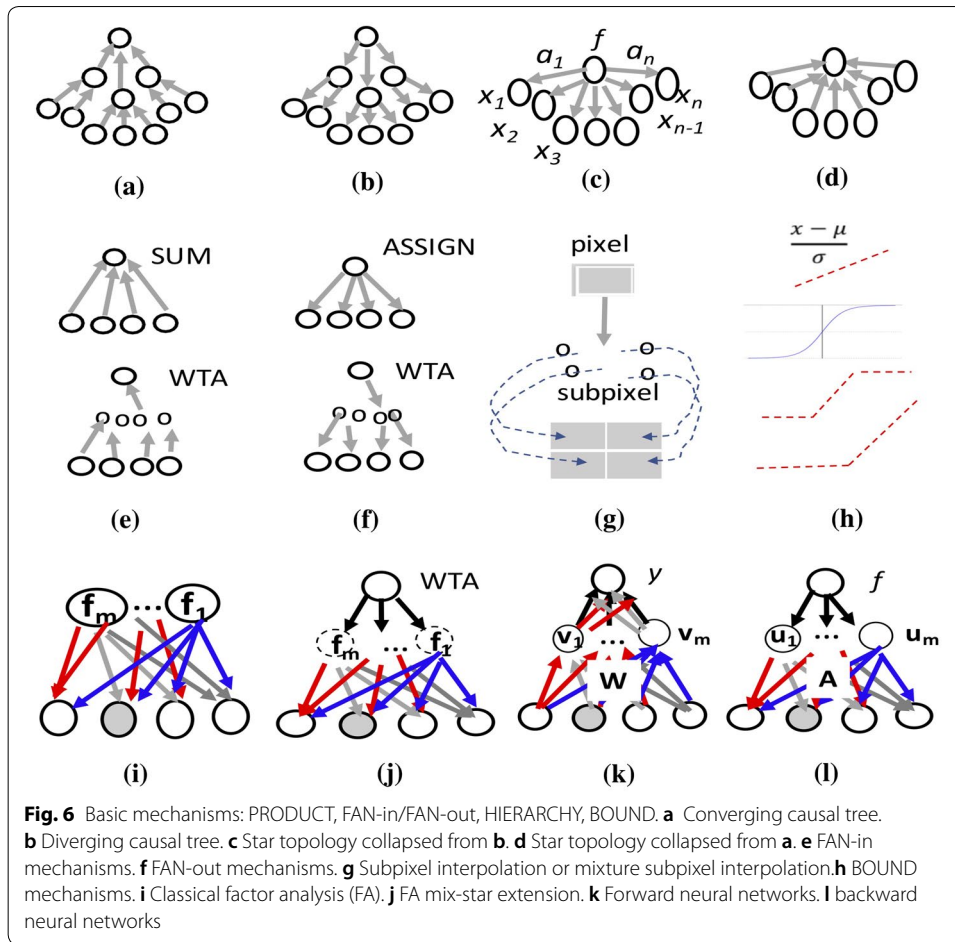
The problem of identifying whether BACK-DOOR DAG is underlying these samples becomes whether the above joint equations are solvable, according to Table 2.

It deserves to recall that Wright's path analysis (Wright 1934) also obtained a set of joint equations like the ones in Eq. (29) or Eq. (33) by considering the second-order statistics along paths among variables in a multiple system from a set of SEM equations. In other words, our approach in this paper inherited the spirit of getting joint equations of the second-order statistics by analysing paths. But there are two major differences in two studies: First, Wright aimed at solving variable quantities based on the joint equations from certain known variables, usually in a small scale system with a few variables. However, our approach aims at using these joint equations to examine whether the topology of a specific DAG is consistent to the one underlying observable data samples, that is, making topology identification in T-phase as summarised in Table 2, while not only parameters and unknowns in the system are re-estimated in P-phase via optimisation for the least error or maximum likelihood but also a best model is searched in C-phase by considering both best-fit and causality. Second, our approach focuses on the relation between ρ -values and the topology in consideration by assuming that variables are normalised to zero means and unit variances. In other words, the joint equations considered in our approach merely involves correlation coefficients which considerably reduce the number of free parameters that are not helpful to topology identification. However, Wright's path analysis considered the joint equations that contains not only correlation coefficients but also path coefficients as well as variances.

Casual ρ -DAG discovery: general case, ρ -SAT problem, and basic mechanisms

We further proceed to the general cases of ρ -DAG, considering the example illustrated in Fig. 5d that shares the same undirected topology illustrated in Fig. 4f. To compute $\rho_{\xi\eta}^o$ from a node ξ to a node η , we again pay attention to those nodes with indegree bigger than 1, which make $\rho_{\xi\eta}^o$ computed from subgraph or subtree instead of simply via a path. For example, the path from X_4 to X_7 contains W_2 with $\deg^-(W_2) = 2$ while one of its two edge comes from a path that contains W_1 with $\deg^-(W_1) = 3$. The information flows that sink at X_7 involve almost the entire DAG except edges $W_4 \rightarrow X_4$ and $W_3 \rightarrow X_5$. Since each of X_1, X_2, X_3 is reachable from X_4 , the information flows come from four sources X_1, X_2, X_3, X_4 instead of merely X_4 . We can compute $\rho_{x_4x_7}^o = EX_4X_7$ in the following hierarchy:

$$[(\rho_{x_1x_4}^o\rho_{x_1w_1} + \rho_{x_2x_4}^o\rho_{x_2w_1} + \rho_{x_3x_4}^o\rho_{x_3w_1})\rho_{w_1w_2} + \rho_{x_4w_3}\rho_{w_3w_2}]\rho_{w_2w_4}\rho_{w_4x_7}, \tag{34}$$



in which three information flows from X_1, X_2, X_3 are merged at W_1 according to Eq. (25), see the above part inside (\dots) , and then further merged with the one from X_4 at W_2 , see the above part inside $[\dots]$. It should be noted that the correlation coefficients $\rho_{x_1x_4}^o, \rho_{x_2x_4}^o, \rho_{x_3x_4}^o$ all have the superscript “o” because the ρ value from X_4 to each of X_1, X_2, X_3 is computed from samples instead of acting as a parameter in the DAG. In a similar way, we may compute $\rho_{x_1x_7}^o, \rho_{x_2x_7}^o, \rho_{x_3x_7}^o$, and thus obtain 4 equations. With X_7 replaced by X_6 and X_5 , we will obtain other 4 + 4 equations. As to each pair among X_5, X_6, X_7 , $\rho_{x_6x_7}^o$ applies to Theorem 3 and thus is simply given by $\rho_{w_4x_7}\rho_{w_4x_6}$, while each of $\rho_{x_5x_7}$ and $\rho_{x_5x_6}$ is not applicable to Theorem 3.

For $\rho_{x_5x_7}$, it contains not just merely a path in the pattern $\leftarrow \leftarrow \dots \leftarrow j \rightarrow \rightarrow \dots \rightarrow$ with $\rho_{x_5w_3}\rho_{w_3w_2}\rho_{w_2w_4}\rho_{w_4x_7}$ and j at W_3 . There is also a flow that injects in at W_2 and thus $\rho_{x_5w_3}\rho_{w_3w_2}$ should be replaced by $[(\rho_{x_1x_4}\rho_{x_1w_1} + \rho_{x_2x_4}\rho_{x_2w_1} + \rho_{x_3x_4}\rho_{x_3w_1})\rho_{w_1w_2} + \rho_{x_5w_3}\rho_{w_3w_2}]$. Similarly we can obtain $\rho_{x_5x_6}$.

Totally, we get 4 + 4 + 4 + 3 = 15 equations jointly for identifying whether the ρ -DAG illustrated in Fig. 5d is underlying a given set of samples by checking whether these joint equations are solvable, for which we are lead to Table 2 again.

In a summary, $\rho_{\xi\eta}^o$ between nodes ξ and η is computed in the following four steps:

Step 1 find out every pair of nodes ξ, η that are directionally correlated by a path in pattern $\leftarrow \leftarrow \dots \leftarrow_j \rightarrow \dots \rightarrow$, where j can locate at ξ or η as well as any middle point;

Step 2 on every path found above, identify those junction nodes featured by their indegrees bigger than 1, and pool all the related paths into a hierarchy similar to the one in Eq. (34).

Step 3 attach each edge $a \rightarrow b$ in this hierarchy by its corresponding ρ_{ab} ;

Step 4 sum up at each junction node the ρ -products of sub-paths from the bottom up, in a way similar to the one in Eq. (34), until the top of the hierarchy.

Alternatively, Eq. (34) can be equivalently rewritten into a format of a summation of four product terms, and each product actually represents a path from X_4 to X_7 .

This is also generally true. There are three possible scenarios for a pair of nodes ξ, η in a ρ -DAG. First, it can be ignored if it is not linked by any path in the pattern $\xi \leftarrow \leftarrow \dots \leftarrow_j \rightarrow \dots \rightarrow \eta$. Second, it is linked by only one such path and thus is applicable to Theorem 3. Third, the pair is linked by a number paths in the pattern $\xi \leftarrow \leftarrow \dots \leftarrow_j \rightarrow \dots \rightarrow \eta$ due to the existence of those junction nodes. For the last two scenarios, we may extend Theorem 3 into the following one.

Theorem 4 *Given a pair of nodes ξ, η in a ρ -DAG, if the pair is linked by a number $n_{\xi\eta}$ paths with each in $\xi \leftarrow x_1^{(r)} \leftarrow \dots \leftarrow x_j^{(r)} \rightarrow \dots \rightarrow x_{m_r}^{(r)} \rightarrow \eta$, $r = 1, \dots, n_{\xi\eta}$ where j may locate at ξ or η as well as any middle point, we have $\rho_{\xi\eta}^o = \sum_{r=1}^{n_{\xi\eta}} \rho_{\xi\eta}^{(r)}$ and $\rho_{\xi\eta}^{(r)} = \rho_{\xi x_1}^{(r)} \rho_{x_1 x_2}^{(r)} \dots \rho_{x_{j-1} x_j}^{(r)} \rho_{x_j x_{j+1}}^{(r)} \dots \rho_{x_{m_r-1} x_{m_r}}^{(r)} \rho_{x_{m_r} \eta}^{(r)}$.*

When $n_{\xi\eta} = 1$, this theorem degenerates back to Theorem 3 that further returns back to Theorem 2 when all the directed edges are replaced by undirected edges. In other words, each directed edge in Theorem 3 and Theorem 4 may be replaced by undirected edge. With one or more of directed edges replaced by undirected edges, Theorem 4 may become applicable to the cases that consider paths with each in the pattern $\xi \overleftarrow{x_1^{(r)}} \overleftarrow{\dots} \overleftarrow{x_j^{(r)}} \overrightarrow{\dots} \overrightarrow{x_{m_r}^{(r)}} \eta$, $r = 1, \dots, n_{\xi\eta}$, where the overline indicates considering not only a case that two nodes are directly linked by either a directed edge or a undirected edge, but also a case that we even do not know whether there is an edge between the two nodes as long as the correlation coefficient between the two nodes can be computed either directly from observable data or indirectly represented by an unknown via a path between the two nodes. Simply, the correlation coefficient is zero if there is no such a path. Therefore, we become able to consider the cases that the directions of edges are partly known and partly unknown.

Further insight on Eq. (34) can be obtained by imagining a special case that each ρ -parameter takes values either around 0 or around 1. Then, a product of two ρ -parameters likes a logical 'AND' gate, while a sum of two ρ -parameters likes a logical 'OR' gate. The problem of identifying ρ -parameters by solving equations with ones like Eq. (34) becomes somewhat similar to the problem of Boolean satisfiability or propositional satisfiability (shortly SAT) that has wide real applications in artificial intelligence, circuit design, and automatic theorem proving (Vizel et al. 2015). On the other hand, we may regard that the classic SAT problem is extended into the above ρ -SAT

problem by solving equations with ones like Eq. (34), from binary valued to real valued and from deterministic to probabilistic. A lot of studies have been made on the classical SAT problem, which may provide some references to further study on the ρ -based SUM-PROD system.

It follows from observing Eq. (34) and Fig. 5d as well as Fig. 6a that SUM operates on a node v with $\deg^-(v) > 1$ and integrates arrows that act in the same dimension or subspace, while PROD operates on an arrow or a directed path of several arrows along a same direction and integrates the arrows that act in different dimension or subspaces with each arrow adding in one dimension. Actually, SUM is of several choices that implements the FAN-in mechanism, as elaborated in Fig. 6e. One end is SUM that treats each fan-in information evenly and thus considers the average of all the fan-in flows, e.g. a classic neuron model, while the other end is WTA (winner-take-all) that treats fan-in flows competitively to pick the best one as winner, e.g. the pooling operation in a conventional neural networks. Between the two ends, SUM may be replaced by some weighted average, and WTA may be replaced by some soft version of competition, e.g. based on a finite mixture.

The information flows in Fig. 6a are converging from the observation nodes to the tree root, featured by $\deg^-(v) > 1$ for every inner node v in the tree. Shortly, we may use a converging tree to name such a tree, which is actually an example of Abstraction model or Yang model. Reversing the direction of every edge, the information flows in Fig. 6b are diverging from tree's root or hidden nodes to the visible nodes, featured by $\deg^+(v) > 1$ for every inner node v in the tree. Shortly, we may use a diverging tree to name such a tree, which is actually an example of Generative model or Ying model.

In Fig. 6b, the FAN-in mechanism is replaced by its counterpart named FAN-out mechanism, as elaborated in Fig. 6f. At one end, the counterpart of SUM is ASSIGN that evenly emits outflows out of every node v with $\deg^+(v) > 1$, e.g. an outer product generative unit in a deconvolution networks or in the reconstruction part of LMSER learning networks (Xu 1993); while the other end is again WTA that emits merely the best one out of all the fan-out flows competitively. Again, there may also be weighted average and soft competition between the two ends,

Particularly, we suggest that such a WTA FAN-out mechanism may improve image reconstruction to perform WTA subpixel interpolation or alternatively mixture subpixel interpolation, as simply sketched in Fig. 6g. The bottom level of reconstruction or deconvolution networks does not terminate at the pixel level. Instead of minimising the error between each pixel and its reconstruction, the bottom level is designed for reconstructing subpixels. For each pixel in a sample image, a WTA competition is made among the reconstructed subpixels underlying the pixel that corresponds to this sample pixel, we minimise the error between this sample pixel and the winning reconstructed subpixel. Alternatively, we may also replace WTA by a soft competition, e.g. by a posterior weighting $p(i|pixel)$ via considering $p(pixel) = \sum_i \alpha_i p(subpixels)$.

Next, the locations of these FAN-in/FAN-out nodes specify a hierarchy of nodes and edges in consideration, which corresponds to a HIERARCHY mechanism that defines a specific combination of those nodes' locations or a partial order hierarchy.

Moreover, all the previous analyses in this paper assume that variables are normalised to zero means and unit variances, which is actually one example of

implementing BOUND mechanism. A variable with bounded variance actually implies that this variable varies with a bounded energy. In computation, requiring a bounded variance is equivalent to requiring a unit variance, implemented simply by normalisation. Equivalently, such a bounded mechanism may be implemented by a nonlinear transform as sketched in Fig. 6h, e.g. sigmoid nonlinearity, its piecewise approximation, and the widely used LUT nonlinearity.

In a summary, the PROD and FAN-in/FAN-out mechanisms jointly describe dependence among variables, with PROD harmonising effects from different parts of one individual and FAN-in/FAN-out mechanisms gathering from and allocating among different individuals; while the HIERARCHY mechanism defines conditional independence and how variables are organised, and the BOUND mechanism ensures practical feasibility. These four basic aspects coordinately operate a casual ρ -DAG model or even a general intelligent system.

Removing HIERARCHY mechanism, e.g. the tree hierarchy in Fig. 6b will collapse such that the tree root becomes the centre of a star topology as illustrated in Fig. 6c, while every path between the tree root and each leaf collapses into an edge of the star. Further removing PROD mechanism, the corresponding ρ product collapses into one variable. Apparently, both the tree hierarchy in Fig. 6b and the star topology in Fig. 6c have the same number of SEM equations and thus a same representative capacity. However, the HIERARCHY and PROD mechanisms impose additionally higher order joint equations on these SEM equations for encoding not only variables but also how they are organised. Similar understandings may be obtained from the converging tree illustrated in Fig. 6a and its collapsed star topology in Fig. 6d.

One step forward from star topology illustrated in Fig. 6c is bundling m star topologies in parallel to share the same n visible nodes, with conditional independence added via m independent root variables. As illustrated in Fig. 6i, it follows from the visible node side that adding an edge between each pair of visible nodes x_i, x_j will result in m loops with each associated with one factor f_r and thus the correlation coefficient ρ_{ij} between the pair is a SUM-PROD form $\rho_{x_i x_j} = \sum_{r=1}^m \rho_{x_i f_r} \rho_{x_j f_r}$. Accordingly, lumping the SEM equations that correspond to the m star topologies together and setting variables in zero means and unit variances, we are actually lead to the following classical factor analysis (FA) model with an m -dimensional factor vector $\mathbf{f} = [f, \dots, f_m]$ of mutually independent elements:

$$\mathbf{x} = \mathbf{A}\mathbf{f} + \mathbf{e}, \mathbf{E}\mathbf{f} = 0, \mathbf{E}\mathbf{e} = 0, \mathbf{E}\mathbf{f}\mathbf{e}^T = 0, \mathbf{E}[\mathbf{f}\mathbf{f}^T] = \mathbf{I}, \mathbf{E}\mathbf{e}\mathbf{e}^T = \Lambda_e \text{ is diagonal}, \quad (35)$$

from which we directly get $\rho_{ii} = 1$ and $[\rho_{ij}] = \mathbf{A}\mathbf{A}^T + \Lambda_e$ as a matrix form of the above SUM-PROD form. It may suffer over-fitting when the number $n \times m$ of unknown parameters in \mathbf{A} is larger than $0.5n(n-1)$. In the degenerated case $m = 1$, we have $[\rho_{ij}] = \mathbf{a}\mathbf{a}^T + \mathbf{I}$ with $\mathbf{a} = [a_1, \dots, a_n]^T$, which actually corresponds to Eq. (13) and may suffer under-fitting because $0.5n(n-1) > n$ for $n > 1$. One way to remedy over-fitting is to prune away extra parameters, which is typically called sparse learning. However, it cannot cover the role of HIERARCHY.

As illustrated in Fig. 6j, we top on the factor \mathbf{f} with another node that exclusively selects one of factors such that visible nodes become no longer simultaneously shared by different stars, which actually acts as a mixture of star topologies that effect exclusively

on visible nodes and thus reduces the coupling width from m towards 1. In general, a HIERARCHY mechanism that defines a multiple level partial order hierarchy via specifying the locations of these FAN-in/FAN-out nodes and their corresponding mechanisms featured by two ends alternatively.

The HIERARCHY mechanism is not just increasing layers to go 'deep'. Multiple linear layers obtained with one layer topped on the other layer will collapse into merely one layer because variables are Gaussians and their summations are still Gaussians. Instead, HIERARCHY defines a partial order hierarchy, which will not collapse because of sparse links and constraints imposed by higher order joint equations that are solvable as described by Table 2. Also, similar arguments are applicable to a neural networks as illustrated in Fig. 6k.

Understandably, the more layers there are, it becomes more easier to accommodate such a hierarchy, which echoes the opinion in Ref. Xu (2017) and interprets why deep learning is preferred, that is, we need to encode not only variables but also how they are organised. In a particular task domain, we only need networks with a limited depth because patterns underlying samples are expressed in hierarchies with a limited depth. Also, a partial order will not be affected by adding one extra edge weighted simply with 1. Thus, increasing depth will not affect performance, but waste more computing cost in both memory and learning time.

Last but not least, the BOUND mechanism not only ensures practical feasibility, but also helps the HIERARCHY mechanism. First, embedding nonlinearity after each summation, i.e., $x^{(i)} = s(\sum_j w_{ij}y^{(j)} + \varepsilon)$, will remedy the above mentioned collapsing of multiple linear layers. Second, it has been discovered in Ref. Xu (1993) that adding a sigmoid nonlinearity after a summation drives hidden nodes towards mutually independent and get organised, which again echoes the opinion made in Ref. Xu (2017) and interprets why it is beneficial to make a bottom-up unsupervised learning as pre-training. Third, a linear transform from one level of variables $\mathbf{y} = \{y^{(j)}\}$ to the next level of variables $\mathbf{x} = \{x^{(i)}\}$ will not only map a specific value of \mathbf{y} into a specific value of \mathbf{x} but also make the neighbourhood or topological relation of \mathbf{y} preserved after being mapped to the one of \mathbf{x} . Thus, we should only consider those of post-summation nonlinearity $s(\cdot)$ that can preserve this nature, e.g. ones in Fig. 6h.

Concluding remarks

Examining AlphaGoZero together with revisiting early studies on A^* search, MCTS is found to share a scouting technique with CNneim-A that was proposed in 1986. The strengths of AlphaGoZero and CNneim-A are further integrated to develop a new family named deep IA-search, including DSA, DCA, DBA, and V-AlphaGoZero, as well as their extensions DSA-E, DCA-E, DBA-E, and AlphaGoZero-E. We are further motivated to perform reasoning with the help of deep IA-search. Especially, casual reasoning is addressed and a correlation coefficient-based approach is proposed for identifying casual ρ -tree and casual ρ -DAG, featured by performing TPC learning to discover causality in three phases. Algorithms are sketched for discovering casual topologies of triplets, stars, trees, and ρ -DAG, with further details on Yule–Simpson's paradox, Pearl's Sprinkler DAG, and Back door DAG. Moreover, the classic Boolean SAT problem is extended

into one ρ -SAT problem, and the roles of four fundamental mechanisms in an intelligent system are elaborated, with insights on integrating these mechanisms to encode not only variables but also how they are organised, as well as on why deep networks are preferred while extra depth is unnecessary.

Also, the insights motivate the possible directions for further investigations:

(a) Though the ρ -based TPC learning proposed only considers the second order statistics, implying that all the variables are Gaussians and all the SEM equations are linear relations, it is directly applicable to tasks with nonGaussian variables and non-linear relations as a sort of approximation. It is likely that the obtained topologies of ρ -tree and ρ -DAG may provide good approximations already, which motivates a direction of extension that performs T-phase as it is but modifies each SEM equation with its linear relation replaced by a post-linear nonlinearity and its driving noise by a nonGaussian variable. Example we may extend Eq. (24) into

$$\begin{aligned} x &= s_{zx}(\rho_{zx}z) + e_{zx}, \quad y = y_z + y_x, \\ y_z &= s_{yz}(\rho_{yz}z) + e_{yz}, \quad y_x = s_{yx}(\rho_{yx}x) + e_{yx}, \end{aligned} \quad (36)$$

where each of $s_{zx}(\cdot)$, $s_{yz}(\cdot)$, $s_{yx}(\cdot)$ is some nonlinear scalar function, e.g. a sigmoid non-linearity, and at least one of e_{zx} , e_{yz} , e_{yx} is a nonGaussian variable. It may be observed that higher components of nonlinear functions get not only each variable but also multiple variables involved in higher order statistics.

(b) Embedding understandings obtained from causal tree into deep learning may improve performances, especially when there is merely a small size of samples. Causal tree or precisely hierarchical SEM equations may be used as an alternative to deep learning. Learning methods may be developed based on Theorem 4 to conduct a constrained optimisation similar to Eq. (20), in a way similar to identifying SPRINKLER DAG and BACK-DOOR DAG. This alternative may also lay a road that turns the black box of deep learning into interpretable causal analyses.

(c) We may further jointly consider a Yang model as illustrated in Fig. 6k and a Ying model as illustrated in Fig. 6l. One early example is bidirectional multilayer neural networks proposed under the name *Lmser* in 1991 (Xu 1991, 1993). Let the output y in Fig. 6k to be directly the input f in Fig. 6l, we are lead to the classical auto-association or autoencoder (Bourlard and Kamp 1988). Differently, *Lmser* extended autoencoder in four aspects. First, the weight matrix A of each layer in Fig. 6k is directly used as the weight matrix W of the corresponding layer in Fig. 6l, simply by letting $A = W^T$, i.e. currently so-called weight sharing or domain transferring. Second, the corresponding nodes are also forced to be same, that is, let $v_i = u_i, i = 1, \dots, m$ as illustrated in Fig. 6k, l. Third, dynamic effect is approximately considered with the reconstructed \bar{x} of the input x by the model in Fig. 6l being re-inputed into the model in Fig. 6k, resulting in a learning rule that consists of a term equivalent to Hinton's wake-sleep algorithm, plus one correcting term that reduces confusion in boundary area. Third, for labeled data, supervised signals may also get the top-down signals with supervised and unsupervised learning jointly (see Section 6 in Ref. Xu (1991)), i.e. currently so-called semi-supervised learning. With the help of these developments, *Lmser* may be used not only

for both reconstruction (i.e. currently so-called data generation) and pattern recognition, but also for concept driven imaginary recall that visualises thinking, pre-activation driven top–down attention, associative memory, pattern transform, and interpreting development of cortical field templates, as well as creative mapping. Moreover, as addressed at the end of the last section in this paper, the nature of preserving the neighbourhood or topological relation by *Lmser* and auto-encoder also facilitates concept forming and organising in the top encoding domain, i.e. the domain of y and f in Fig. 6k, l, which is superior to those models in lack of such preservation, e.g. variational autoencoder (Schmidhuber 2015), deep generative model (Rezende et al. 2016), generative adversarial networks (Goodfellow et al. 2014). Furthermore, further improvement may be developed by RPCL-*Lmser* and LVQ-*Lmser* that perform vector quantisation in the encoding domain of *Lmser* by LVQ (Kohonen 1995) for labelled data and RPCL (Xu et al. 1993) for unlabelled data.

Authors' contributions

All from the sole author. The author read and approved the final manuscript.

Author details

¹ Centre for Cognitive Machines and Computational Health (CMaCH), The School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, SEIEE Building 3, 800 Dongchuan Road, Minhang District, Shanghai 200240, China. ² Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.

Acknowledgements

This work was supported by the Zhi-Yuan chair professorship start-up Grant (WF220103010) from Shanghai Jiao Tong University.

Competing interests

The author declares that there is no competing interests.

Availability of data and materials

Not applicable.

Ethics approval and consent to participate

Not applicable.

Funding

WF220103010, Shanghai Jiao Tong University.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 16 March 2018 Accepted: 10 September 2018

Published online: 29 September 2018

References

- Anderson TW, Rubin H (1956) Statistical inference in factor analysis. In: Proceedings of the third Berkeley symposium on mathematical statistics and probability, vol. 5, pp 111–150
- Bouillard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. Biol Cybernet 59(4–5):291–294
- Browne CB, Powley E, Whitehouse D, Lucas SM, Cowling PI, Rohlfshagen P, Tavener S, Perez D, Samothrakis S, Colton S (2012) A survey of monte carlo tree search methods. IEEE Trans Comput Intell AI Games 4(1):1–43
- Cai R, Zhang Z, Hao Z (2013) Sada: a general framework to support robust causation discovery. In: International conference on machine learning, pp 208–216
- Clark C, Storkey A (2015) Training deep convolutional neural networks to play go. In: International conference on machine learning, pp 1766–1774
- Ferrucci D, Brown E, Chu-Carroll J, Fan J, Gondek D, Kalyanpur AA, Lally A, Murdock JW, Nyberg E, Prager J (2010) Building watson: an overview of the deepqa project. AI Magaz 31(3):59–79
- Ferrucci D, Levas A, Bagchi S, Gondek D, Mueller ET (2013) Watson: beyond jeopardy!. Artif Intell 199:93–105
- Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

- Hart T, Edwards D (1961) The alpha-beta heuristic
- Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans Syst Sci Cybern* 4(2):100–107
- Hoyer PO, Janzing D, Mooij JM, Peters J, Schölkopf B (2009) Nonlinear causal discovery with additive noise models. In: *Advances in neural information processing systems*, pp 689–696
- Kocsis L, Szepesvári C (2006) European conference on machine learning. Bandit based monte-carlo planning. Springer, Berlin, pp 282–293
- Kohonen T (1995) Self-organizing maps. Learning vector quantization. Springer, Berlin, pp 175–189
- Land AH, Doig AG (1960) An automatic method of solving discrete programming problems. *Econometrica* 28:497–520
- LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436
- Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540):529
- Pearl J (1984) *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc, Reading
- Pearl J (1986) Fusion, propagation, and structuring in belief networks. *Artif Intell* 29(3):241–288
- Pearl J (1988) *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Mateo
- Pearl J (2010) An introduction to causal inference. *Int J Biostat* 6(2):1–62
- Pearl J, Verma TS (1991) Equivalence and synthesis of causal models. In: *Proceedings of sixth conference on uncertainty in artificial intelligence*, pp 220–227
- Reichenbach H (1956) *The direction of time*. University of California Press, Berkeley
- Rezende DJ, Mohamed S, Danihelka I, Gregor K, Wierstra D (2016) One-shot generalization in deep generative models. [arXiv: 1603.05106](https://arxiv.org/abs/1603.05106)
- Rubin DB (1974) Estimating causal effects of treatments in randomized and nonrandomized studies. *J Educ Psychol* 66(5):688
- Rubin DB, John L (2011) Rubin causal model. In: *International encyclopedia of statistical science*. Springer, Berlin, pp 1263–1265
- Schmidhuber J (2015) Deep learning in neural networks: an overview. *Neural Netw* 61:85–117
- Shimizu S, Hoyer PO, Hyvärinen A, Kerminen A (2006) A linear non-Gaussian acyclic model for causal discovery. *J Mach Learn Res* 7:2003Machine Learning Res–2030
- Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M (2016) Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489
- Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A (2017) Mastering the game of go without human knowledge. *Nature* 550(7676):354
- Spearman C (1904) "General intelligence", objectively determined and measured. *Am J Psychol* 15(2):201–292
- Spirtes P, Glymour C (1991) An algorithm for fast recovery of sparse causal graphs. *Soc Sci Comput Rev* 9(1):62–72
- Spirtes PG, Glymour C (1993) Causation, prediction and search. *Lecture Notes in Statistics*. vol. 81. Springer, Berlin
- Spirtes P, Glymour CN, Scheines R (2000) *Causation, prediction, and search*. MIT Press, New York
- Sutton RS, Barto AG (1998) *Reinforcement learning: an introduction*, vol 1. MIT Press, Cambridge
- Sutton RS, McAllester DA, Singh SP, Mansour Y (2000) Policy gradient methods for reinforcement learning with function approximation. In: *Advances in neural information processing systems*, pp 1057–1063
- Vizel Y, Weissenbacher G, Malik S (2015) Boolean satisfiability solvers and their applications in model checking. *Proc IEEE* 103(11):2021–2035
- Watkins CJ, Dayan P (1992) Q-learning. Springer, Berlin, pp 279–292
- Wright S (1921) Correlation and causation. *J Agric Res* 20(7):557–585
- Wright S (1934) The method of path coefficients. *Ann Math Stat* 5(3):161–215
- Xu L (1986a) A note on a new heuristic search technique algorithm sa. In: *Proc. of 8th international conference on pattern recognition*, vol. 2. IEEE Press, Paris, pp 992–994
- Xu L (1986b) Investigation on signal reconstruction, search technique, and pattern recognition. PhD dissertation. Tsinghua University, Tsinghua
- Xu L (1987) Can sa beat the exponential explosion? In: *Proc. of 2nd international conference on computers and applications*. IEEE Press, Beijing, pp 706–713
- Xu L (1991) Least mse reconstruction for self-organization:(i) multi-layer neural nets and (ii) further theoretical and experimental studies on one layer nets. In: *Proceedings of the international joint conference on neural networks*. Singapore, pp 2363–2373
- Xu L (1993) Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Netw* 6(5):627–648
- Xu L (1995) Bayesian-kullback coupled ying-yang machines: unified learnings and new results on vector quantization. In: *Proceeding international conference neural information process (ICONIP '95)*. Publishing House of Electronics Industry, Beijing, pp 977–988
- Xu L (1996) A unified learning scheme: Bayesian–Kullback ying-yang machine. In: *Advances in neural information processing systems*, pp 444–450
- Xu L (2010) Bayesian Ying–Yang system, best harmony learning, and five action circling. *Front Elect Eng China* 5(3):281–328
- Xu L (2017) The third wave of artificial intelligence. *KeXue* 69(3):1–5 (in Chinese)
- Xu L, Pearl J (1987) Structuring causal tree models with continuous variables. In: *Proceedings of the 3rd annual conference on uncertainty in artificial intelligence*, pp 170–179
- Xu L, Yan P, Chang T (1987) Algorithm cnneim-a and its mean complexity. In: *Proc. of 2nd international conference on computers and applications*. IEEE Press, Beijing, pp 494–499
- Xu L, Yan P, Chang T (1988) Best first strategy for feature selection. In: *Proc. of 9th international conference on pattern recognition*, vol. 2. IEEE Press, Rome, pp 706–709

- Xu L, Yan P, Chang T (1989) Application of state space heuristic search technique in pattern recognition. *Comput Appl Softw* 6(1):27–34
- Xu L, Krzyzak A, Oja E (1993) Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. *IEEE Trans Neural Netw* 4(4):636–649
- Zhang B, Zhang L (1985) A new heuristic search technique-algorithm sa. *IEEE Trans Pattern Anal Mach Intell* 1:103–107
- Zhang K, Hyvärinen A (2009) On the identifiability of the post-nonlinear causal model. In: *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, pp 647–655

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
