Learning Functional Dependency Networks based on Genetic Programming

Wing-Ho Shum, Kwong-Sak Leung
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong
{whshum, ksleung}@cse.cuhk.edu.hk

Man-Leung Wong
Dept. of Computing & Decision Sciences
Lingnan University
Hong Kong
mlwong@ln.edu.hk

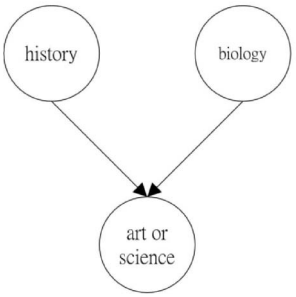
Abstract

Bayesian Network (BN) is a powerful network model, which represents a set of variables in the domain and provides the probabilistic relationships among them. But BN can handle discrete values only; it cannot handle continuous, interval and ordinal ones, which must be converted to discrete values and the order information is lost. Thus, BN tends to have higher network complexity and lower understandability. In this paper, we present a novel dependency network which can handle discrete, continuous, interval and ordinal values through functions; it has lower network complexity and stronger expressive power; it can represent any kind of relationships; and it can incorporate a-priori knowledge though user-defined functions. We also propose a novel Genetic Programming (GP) to learn dependency networks. The novel GP does not use any knowledge-guided nor application-oriented operator, thus it is robust and easy to replicate. The experimental results demonstrate that the novel GP can successfully discover the target novel dependency networks, which have the highest accuracy and the lowest network complexity.

1 Introduction

BN has been applied to different areas [8]. A good BN should balance the accuracy and the network complexity, which is defined in term of the total number of entries in the conditional tables [9]. Variables can be classified into discrete, continuous, interval and ordinal variables. Since BN can only handle discrete values, continuous, interval and ordinal values must be discretized and thus the order information is ignored [12]. The disability to handle continuous, interval and ordinal values increases the network complexity and the relationships represented by BN become less understandable.

Consider a university programme selection problem that has two ordinal variables. Suppose a high school student would choose a science programme if he/she got a better grade in biology; he/she would study an art programme if he/she done better in history; otherwise, he/she would study anyone randomly. Figure 1 shows the BN representing the problem. Since BN cannot compare the course grades directly, it enumerates all instances of the combination of the subject grades and calculates the corresponding probabilities. Although there are only two subjects, the conditional tables are large and have a lot of entries, i.e. the network complexity is high and the meanings of the relationships are unclear and incomprehensible. The BN shows there are some relationships, but it cannot illustrate that the grade comparison result affects the programme selection. The understandability of the relationships is reduced because there are several entries in the conditional tables.



history	A	B	C	D	E	F
	0.1	0.2	0.2	0.2	0.2	0.1

biology	A	B	C	D	E	F
	0.1	0.2	0.2	0.2	0.2	0.1

		art or science	
history	biology	art	science
A	A	0.5	0.5
A	B	0.8	0.2
A	C	0.8	0.2
A	D	0.8	0.2
A	E	0.8	0.2
A	F	0.8	0.2
B	A	0.2	0.8
B	B	0.5	0.5
B	C	0.8	0.2
B	D	0.8	0.2
B	E	0.8	0.2
B	F	0.8	0.2
C	A	0.2	0.8
C	B	0.2	0.8
C	C	0.5	0.5
C	D	0.8	0.2
C	E	0.8	0.2
C	F	0.8	0.2
D	A	0.2	0.8

Figure 1. The Bayesian Network for the programme selection problem.

Continuous BN is proposed to handle continuous values

Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM'05)
1550-4786/05 $20.00 © 2005 IEEE

[5]. They rely on the assumption of parametric or semi-parametric families, like Gaussian distributions. But continuous BN cannot handle discrete values. Mixed BN tries to handle both of them, but it cannot represent discrete nodes with continuous parents [3].

We propose a novel dependency network, Functional Dependency Network (FDN) to incorporate functions into BN. Besides of the relationships among variables, it can also represent the ones among functions of variables. Any kind of functions can be used and different relationships can be represented. They can handle discrete, continuous, interval and ordinal values, thus the order information is preserved. They could be simply a value comparison among variables; they could be any kind of calculations among variables; they could represent any kind of special relationships among variables; and they could incorporate a-priori knowledge as user-defined functions. The functions increase the network's understandability and strengthen its expressive power.

Figure 2 shows the FDN for the programme selection problem. The FDN has a functional node, $history > biology$. It represents the grade comparison between the two subjects and its conditional table has only one entry specifying the function $>$. $>$ returns 1 if the first argument is greater than the second one; if the two arguments are equal, it returns 0; otherwise, -1 is returned. With the functional node, the FDN has reduced the number of entries in the conditional tables from $6 + 6 + 6^2 * 2$, i.e. 84 to $6 + 6 + 3 * 2 + 1$, i.e. 19, the network complexity is significantly reduced. By realizing the meaning of $>$, it is easy to understand and interpret the meanings of the relationships; if necessary, the relationships can be expressed in rule format easily. For instance, the rules could be merged into an expert system and the number of rules is proportional to the number of entries in the conditional tables.

Wong and Leung proposed an evolutionary programming (EP) to learn BN [15]. Myers et al. used a genetic algorithm (GA) to learn BN from incomplete data [11]. However, these methods are not flexible enough to represent functions and thus they cannot learn FDNs.

GP is a branch of evolutionary computation (EC). It has greater representation power than EP and GA. GP has been applied to different areas, like shortest path finding and classification [4]. Wong and Leung proposed a grammar-based GP to handle the closure problem and incorporate a-priori knowledge [14].

In this paper, we present a novel GP, MDL Genetic Programming (MDLGP) to learn FDNs. Unlike other EC algorithms for BN learning, MDLGP does not use any knowledge-guided nor application-oriented operator, thus it is robust and easy to replicate. The paper is organized as follows. We introduce the FDN in next section, followed by a description of the MDLGP. The experimental results

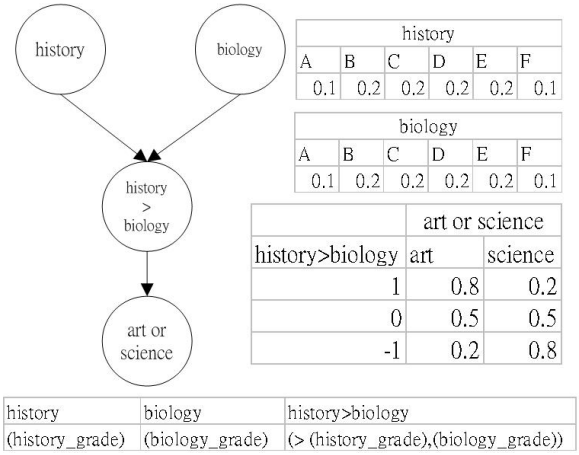


Figure 2. The Functional Dependency Network for the programme selection problem.

are presented in Section 3. A conclusion is given in the last section.

2 The Proposed Algorithm

2.1 The Functional Dependency Network

BN is a directed acyclic network. Figure 1 shows an example of BN. A node denotes a variable in the domain and a directed link, $N_i \rightarrow N_j$ represents the dependency relationships between the child N_j and the parent N_i. Each variable node has a conditional table specifying the probability of each particular value of the variable node given an instantiation of the parents. In other words, for each variable node N_i with parents Υ_{N_i}, there is a conditional table specifying the conditional probability distribution $P(N_i|\Upsilon_{N_i})$; for each N_i with no parent, the conditional table specifying the priori probability distribution $P(N_i)$. BN encodes the joint probability distribution of the domain variables, $U = \{N_1, ..., N_n\}$

$$P(N_1, ..., N_n) = \prod_i P(N_i|\Upsilon_{N_i}) \quad (1)$$

The FDN is a directed acyclic network. Figure 2 shows an example of the FDN. A new type of node is introduced, functional node, it represents functions of variables. The functions can have any number of arguments and any number of nesting levels. Their conditional tables have only one entry, which specifies the functions producing the value of the functional node given an instantiation of its parents.

Like GP, the function set is defined by the user. Different types of functions operate on different types of variables.

For example, it is prohibited to apply the continuous function log to a discrete variable.

Similar to BN, a variable node can handle discrete values only. If a continuous functional node has a variable node as its child, discretization is needed.

A continuous variable node can either have another variable node or a continuous functional node or both as its children. If the child is a variable node, discretized values are produced; if the child is a continuous functional node, continuous values are generated according to a Gaussian distribution function. Each entry in the conditional table represents an interval. To generate a continuous value, 1) select an entry according to the probability, 2) according to Gaussian distribution function, pick up a continuous value randomly given the mean and the standard deviation of the interval.

Although the functions may slightly increase the computation complexity, they could reduce the network complexity and save the time in searching the large conditional tables. In other words, the overall performance of the FDN is similar to (or even smaller than) that of BN. Moreover, the relationships described by the functions are also more understandable than those represented by a number of entries in the conditional tables.

2.2 The MDL Genetic Programming

2.2.1 The Population

We propose a novel GP, the MDLGP to learn the FDN. It has a population of individuals and each individual represents one FDN. Individuals are of the form, $(< parents >\rightarrow< child >)_1.....(< parents >\rightarrow< child >)_y$ where $y \in Z+$. We call each of the $(< parents >\rightarrow< child >)$ as a fragment, which represents the relationships between the $< parents >$ and the $< child >$. $< parents >$ and $< child >$ are in the prefix form. $< parents >$ denotes one or more parents and a parent can either be a variable node or a functional node. $< child >$ is a variable node. Different fragments can have the same $< child >$. The fragments containing functional nodes also represent the variables in the functions. For instance, the individual representing the FDN in Figure 2 is $((> history\ biology) \rightarrow art\_or\_science)$

The MDLGP has great representation power. A FDN can be represented by more than one instantiations of individual. For example, both of the individuals $(variable\_A,\ variable\_B \rightarrow variable\_C)$ and $(variable\_A \rightarrow variable\_C)(variable\_B \rightarrow variable\_C)$ represent the same FDN. Any number of nesting levels is also possible.

The MDLGP uses a grammar to prevent the closure problem, i.e. the type mismatching issue among functions and variables. Table 1 shows the grammar.

Table 1. The grammar for the MDLGP.

network := link*
link := (parents → child)
parents := parent*
parent : = node || functional_node
child := *any_variable*
node := *any_variable*
functional_node := discrete_fnode || continuous_fnode || ordinal_fnode|| interval_fnode
discrete_fnode := d_comparison
continuous_fnode := c_comparison || c_operation
ordinal_fnode := o_comparison
interval_fnode := i_operation
d_comparison := (> *discrete_variable discrete_variable*)
c_comparison := (> *continuous_variable continuous_variable*)|| (> c_operation *continuous_variable*)|| (> c_operation c_operation)
o_comparison := (> *ordinal_variable ordinal_variable*)
c_operation := (coperator *continuous_variable continuous_variable*)
i_operation := (ioperator *interval_variable interval_variable*)
coperator := + || - || * || /
ioperator := + || -
where > returns 1 if the first argument is larger than the second one; it returns 0 if they are equal; else it returns -1 and * denotes one or more occurrences

The MDLGP translates an individual into a network fragment by fragment. Individuals may carry invalid fragments, which would create cycles in the network. The MDLGP validates them one by one, starting from the leftmost one. If the fragment is valid, it would be translated as links and nodes into the network. If it would create cycle in the network, it would be simply ignored. The advantages of simply ignoring the invalid fragments, instead of repairing or preventing them are, 1) a fragment may be invalid in the current instantiation of individual, but may be valid and contributive while it is replicated to the other through genetic operators; 2) the crossover can be used. It is an important genetic operator which can promote the convergence of EC; 3) no repairing nor preventing heuristic is required. Different applications may need different heuristics and it is hard to define a good one. A bad heuristic would heavily degrade the performance; 4) the diversity of the population is promoted. If the invalid fragments are repaired or prevented, it may reduce the diversity of the population, which may in turn degrade the performance of the MDLGP.

The population has a fixed number of individuals and the initial population is generated randomly.

2.2.2 Extended MDL Calculation

There are many different scoring schemes to evaluate BN. They measure the fitness of BN, in terms of the accuracy and the network complexity. They are either derived from Bayesian statistics, information theory or MDL principle [10].

The MDLGP uses MDL to evaluate the fitness of individuals. MDL balances between the network's accuracy and simplicity; and it is a combination of the network description length and the data description length. The smaller MDL score is the better.

We extend MDL to incorporate the functional nodes. Let G be a network, $U = \{N_1, ..., N_n\}$ be the variables in the domain, $V = \{F_1, ..., F_f\}$ be the functional nodes in G, Υ_ϕ be the parents of the variable node or the functional node ϕ, s_ϕ be the number of possible states of the variable node or the functional node ϕ and d is the number of bits required to store a numerical value. The MDL score of G is the sum of the MDL scores of U and V.

$$MDL(G) = \sum_i MDL(N_i, \Upsilon_{N_i}) + \sum_j MDL(F_j, \Upsilon_{F_j}) \quad (2)$$

a variable node or a functional node ϕ's MDL score is given by

$$MDL(\phi, \Upsilon_\phi) = ND(\phi, \Upsilon_\phi) + DD(\phi, \Upsilon_\phi) \quad (3)$$

where ϕ is either N_i or F_j

The network description length measures the number of bits encoding the network. To represent a particular FDN, the following information is necessary and sufficient:
• A list of the parents of each variable node and the number of bits required is $|\Upsilon_{N_i}| log_2(n + f)$.
• The set of conditional probabilities associated with each variable node and the number of bits required is $d(s_{N_i} - 1) \prod_{\varphi \in \Upsilon_{N_i}} s_\varphi$.
• The functional description associated with each functional node. For instance, the functional description of the functional node $(> variable\_1\ variable\_3)$ is $> v1v3$ and the length is $5 * 8$, i.e. 40 bits.

The network description length of a node, $ND(\phi, \Upsilon_\phi)$ is

$$ND(\phi, \Upsilon_\phi) = \begin{cases} fdl(\phi) & \text{if } \phi \text{ is } F_j \\ |\Upsilon_\phi| log_2(n + f) + d(s_\phi - 1) \prod_{\varphi \in \Upsilon_\phi} s_\varphi & \text{else} \end{cases} \quad (4)$$

where $fdl()$ is the length of the functional description and is measured in term of bits

The data description length measures the number of bits encoding the data set given the network, using Huffman code and the probabilities of occurrences of each instantiation in the data set. Closer the probabilities represented by the network to the correct ones smaller the data description length, i.e. the data description length reflects the network accuracy. Functional nodes' data description lengths are counted as 0, because 1) they simply apply the functions on the parents' instantiations and the relationships are absolutely certain (according to the MDL principle, the data description length would be calculated as 0 if the relationship is certain); 2) the calculation results are not actually encoded in the data set.

The data description length of a node, $DD()$ is

$$DD(\phi, \Upsilon_\phi) = \begin{cases} 0 & \text{if } \phi \text{ is } F_j \\ \sum_{\phi, \Upsilon_\phi} M(\phi, \Upsilon_\phi) log_2 \frac{M(\Upsilon_\phi)}{M(\phi, \Upsilon_\phi)} & \text{else if } \Upsilon_\phi \neq \{\} \\ \sum_{\phi, \Upsilon_\phi} M(\phi) log_2 \frac{e}{M(\phi)} & \text{else} \end{cases} \quad (5)$$

where $M()$ is the number of data items that match a particular instantiation in the data set and e is the total number of data items in the data set (the log_2 function will be 0 if $M(\Upsilon_\phi)$ is 0).

Sine the MDL calculation is time consuming, the MDLGP uses a multi-level hash table to store the calculation results, $MDL(\phi, \Upsilon_\phi)$. When it encounters the same ϕ and Υ_ϕ, it simply retrieves the stored result, instead of performing the calculation again.

2.2.3 Selection and Reproduction

The MDLGP selects individuals for reproduction through tournament competition. Each individual competes with a number of randomly chosen individuals. According to the number of winning competitions, fitter individuals are selected.

The MDLGP has four genetic operators, they are the mutation, the crossover, the insertion and the deletion. The mutation and the crossover are the canonical ones of GP. The insertion and the deletion are novel genetic operators, the insertion inserts a randomly generated fragment or a parent into a random position of the selected individual; the deletion deletes a randomly chosen fragment or a parent from the selected individual.

After the reproduction, the total number of individuals and offspring is double. To keep the population size remain constant, the worst half of them are destroyed.

Then, the extinction is used to further promote the diversity of the population [6]. Fitter individuals have higher chance to reproduce more offspring and their offspring also have higher chance to survive. As the learning process goes on, the individuals in the population become similar with each other, i.e. the diversity is decreased. The extinction promotes the diversity by replacing the worst portion of the population with randomly generated individuals.

Once the maximum number of generations is met, the learning process is stopped and the fittest individual in the population is chosen as the final solution.

Table 2. The pseudocode used to generate Data Set 1.

```
history_test1 = random(1000);
history_test2 = random(1000);
biology_test1 = random(1000);
biology_test2 = random(1000);
if (history_test1+history_test2>160)
    history = A;
else if (history_test1+history_test2>120)
    history = B;
else if (history_test1+history_test2>80)
    history = C;
else if (history_test1+history_test2>40)
    history = D;
else
    history = E;
if (biology_test1+biology_test2>160)
    biology = A;
else if (biology_test1+biology_test2>120)
    biology = B;
else if (biology_test1+biology_test2>80)
    biology = C;
else if (biology_test1+biology_test2>40)
    biology = D;
else
    biology = E;
if (history>biology)
   art_or_science = art (p = 0.8);
   art_or_science = science (p = 0.2);
else if (history<biology)
   art_or_science = science (p = 0.8);
   art_or_science = art (p = 0.2);
else
   art_or_science = art (p = 0.5);
   art_or_science = science (p = 0.5);

where random() returns a random real number
and p is the probability.
```

3 Experimental results

3.1 Data Sets

We have evaluated the MDLGP and the FDN on one synthetic and two real-life data sets. Table 2 shows the pseudocode used to generate the synthetic data set, Data set 1, that consists of continuous, ordinal and discrete variables. The data set has 7 variables ($history\_test1$, $history\_test2$, $biology\_test1$, $biology\_test2$, $history$, $biology$ and $art\_or\_science$) and 1000 data items. It simulates a real-life situation in a school. Suppose there are two subjects, history and biology. Each subject has two tests and the sum of the marks determines the course grade, from A to F. The grade comparison result between the subjects determines if the student would choose a science or an art programme in university.

Data set 2 is a real-life data set that models the psychological experiments reported by Siegler [13]. It is available from UCI machine learning repository [7]. It has 5 continuous variables and 625 data items. Each data item is classified as having the balance scale tip to the right, tip to the left, or be balanced. The variables are $left\_distance$, $left\_weight$, $right\_distance$ and $right\_weight$. The correct way to find the class is the greater of $(left\_distance * left\_weight)$ and $(right\_distance * right\_weight)$. If they are equal, it is balanced.

Data set 3 is the Monk data set from UCI machine learning repository [7]. It has 7 nominal variables and 556 data items. The variables are $variable\_0$, $variable\_1$, $variable\_2$, $variable\_3$, $variable\_4$, $variable\_5$ and $variable\_6$. The relationships, $variable\_0 = variable\_1$ and $variable\_4 = 1$ determine the values of $variable\_6$.

3.2 Evaluation Methods

In the experiments, the MDLGP has learnt both of the FDN and BN. They are compared with Belief Network PowerConstructor (PC), WinMine (WM) and B-Course (BC) [1, 2, 12]. The PC and the WM are deterministic algorithms. The experimental results are evaluated in terms of,

- MDL, the average and the best MDL scores of the final solutions, the smaller the better,
- EN, the average and the best total number of entries in the conditional tables of the final solutions, the smaller the lower network complexity,
- DDL, the average and the best data description length of the final solutions, the smaller the more accurate,
- the average and the best structural differences of variable nodes, which are measured in terms of the number of links inserted (LI), the number of links omitted (LO) and the number of links reversed (LR) between the original structure and the final solutions,
- the average and the best structural differences of functional nodes, which are measured in terms of the number of functional nodes inserted (FI) and the number of functional nodes omitted (FO) between the original structure and the final solutions.

The experimental results are obtained from 10 independent runs.

3.3 Results for Data Set 1

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the

Table 3. Results for Data Set 1.

	MDL	EN	DDL
MDLGP with FDN	10858.7 (10858.7)	79.0 (79.0)	10639.4 (10639.4)
MDLGP with BN	12076.7 (12076.7)	280.0 (280.0)	11143.3 (11143.3)
PC	12497.9 (12497.9)	440.0 (440.0)	11018.0 (11018.0)
WM	12114.0 (12114.0)	320.0 (320.0)	11088.2 (11088.2)
BC	12125.2 (12125.2)	290.0 (290.0)	11114.4 (11114.4)

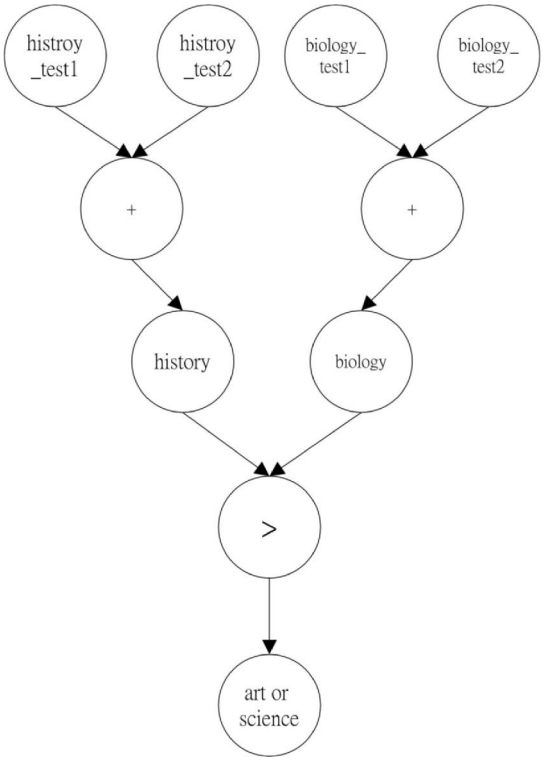


Figure 3. The network learnt by the MDLGP with the FDN for Data Set 1.

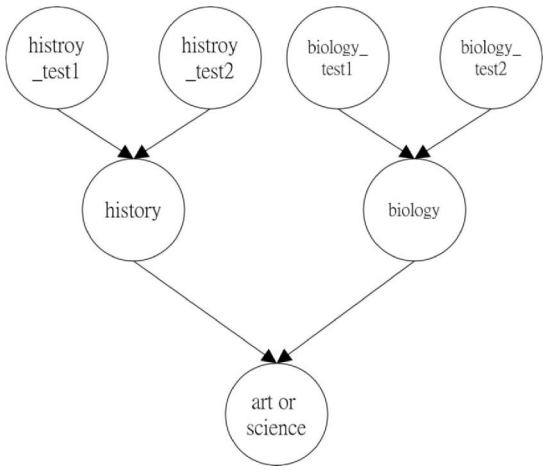


Figure 4. The network learnt by the WM for Data Set 1.

Table 4. Results for Data Set 2.

	MDL	EN	DDL
MDLGP with FDN	5923.9 (5850.2)	31.6 (30.0)	5870.5 (5804.8)
MDLGP with BN	6491.8 (6491.8)	65.0 (65.0)	6290.3 (6290.3)
PC	11038.4 (11038.4)	1895.0 (1895.0)	5804.8 (5804.8)
WM	6614.1 (6614.1)	105.0 (105.0)	6348.0 (6348.0)
BC	6491.8 (6491.8)	65.0 (65.0)	6290.3 (6290.3)

deletion rate are 1000, 7, 50, 0.33, 5, 0.3, 0.3, 0.3 and 0.1 respectively.

Table 3 shows the results for Data Set 1. The values in parenthesis are the best results of the runs. Since the PC and the WM are deterministic algorithms, they always got the same results. The MDLGP with the FDN and the MDLGP with BN always converged. Figures 3 and 4 show the networks learnt by the MDLGP with the FDN and the WM respectively. The MDLGP with the FDN has got the smallest values of MDL, EN and DDL. It has learnt the network correctly, which has the highest accuracy and the smallest number of entries in the conditional tables. Among the other algorithms, the WM has learnt the most similar and accurate network without functional nodes, but it has got much more entries in the conditional tables. More entries reduce the understandability of the relationships. Although the WM has found the dependency relationships, it cannot illustrate their meanings. With the functional nodes, the MDLGP with the FDN has reduced the number of entries from 320 to 79 and the FDN provided the meanings of the relationships as well.

3.4 Results for Data Set 2

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the deletion rate are 1000, 7, 50, 0.5, 5, 0.3, 0.1, 0.3 and 0.3 respectively.

Table 4 shows the results for Data Set 2. Figures 5 and 6 show the networks learnt by the MDLGP with the FDN and the PC respectively. The best network from the MDLGP with the FDN has got the smallest values of MDL and EN. It represents Data set 2 accurately and has had the smallest number of entries in the conditional tables. Among the others, the PC has learnt the most similar network without functional nodes, it has had the smallest value of DDL. Although its network represents Data set 2 accurately, it has got a lot of entries in the conditional tables, thus the mean-

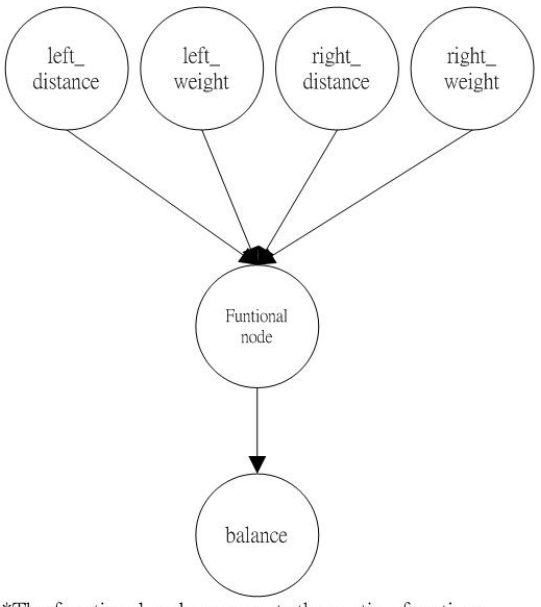


Figure 5. The network learnt by the MDLGP with the FDN for Data Set 2.

Table 5. Results for Data Set 3.

	MDL	EN	DDL
MDLGP with FDN	4911.4 (4911.4)	34.0 (34.0)	4867.0 (4867.0)
MDLGP with BN	5037.2 (5037.2)	89.0 (89.0)	4881.0 (4881.0)
PC	5169.3 (5169.3)	66.0 (66.0)	4993.7 (4993.7)
WM	5272.3 (5272.3)	25.0 (25.0)	5253.0 (5253.0)
BC	5037.2 (5037.2)	89.0 (89.0)	4881.0 (4881.0)

ings of the relationships are unclear and incomprehensible. In contrast, the MDLGP with the FDN has significantly reduced the number of entries from 1895 to 30 and the functional node also illustrated the meanings of the relationships.

3.5 Results for Data Set 3

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the deletion rate are 1000, 7, 50, 0.5, 5, 0.2, 0.3, 0.2 and 0.2 respectively.

Table 5 shows the results for Data Set 3. Figures 7 and 8 show the networks learnt by the MDLGP with the FDN, the MDLGP with BN and the BC respectively. Both of the MDLGP with the FDN and the MDLGP with BN are always converged. The MDLGP with the FDN has got the smallest values of MDL and DDL. It has learnt the correct network which has the highest accuracy and the second smallest number of entries in the conditional tables. The MDLGP with BN and the BC have achieved the same result, they have got the second smallest values of MDL and DDL, but the worst in EN; they have learnt the most similar network without functional node. Their networks infer the values of $variable\_6$ by enumerating the instances of the combination of $variable\_0$, $variable\_1$ and $variable\_4$, thus the conditional tables are large. In contrast, the MDLGP with the FDN has produced a concise and understandable network, which illustrated the values of $variable\_6$ are determined by the relationships $variable\_0 = variable\_1$ and $variable\_4$. Although the WM has got the smallest value of EN, it has had the worst result in MDL and DDL, i.e. it has learnt the simplest but the most inaccurate network.

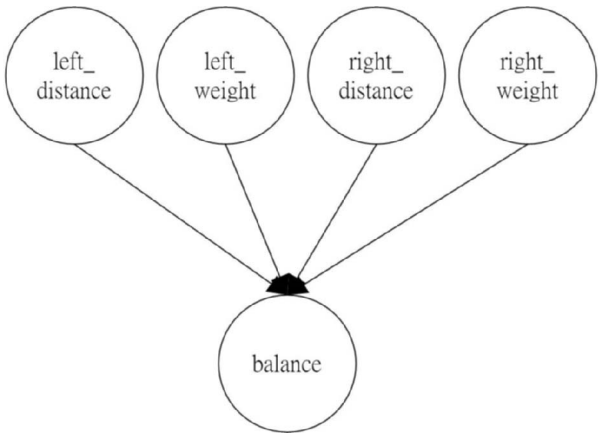


Figure 6. The network learnt by the PC for Data Set 2.

4 Conclusion

In this paper, we propose a novel dependency network with functions, the FDN. It can handle discrete, continuous, interval and ordinal values; it can represent any kind of relationships; and it can incorporate a-priori knowledge as user-defined functions. The FDN has lower network complexity and stronger expressive power than BN. We also present a novel GP, the MDLGP to learn the FDN. The MDLGP with the FDN is evaluated and compared with several algorithms. The experimental results demonstrate that the MDLGP can successfully discover the target FDNs, which have the highest accuracy and the lowest network complexity.

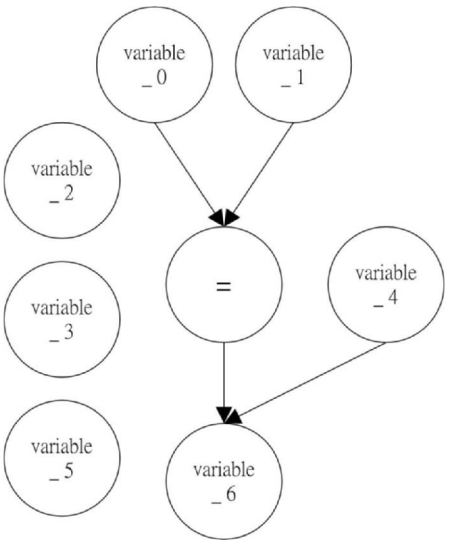


Figure 7. The network learnt by the MDLGP with the FDN for Data Set 3.

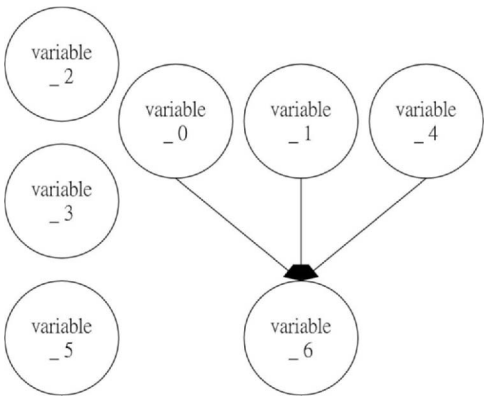


Figure 8. The network learnt by the MDLGP with BN and the BC for Data Set 3.

Acknowledgment

This work is partially supported by the Earmarked Grant LU 3012/01E of Hong Kong SAR.

References

[1] Jie Cheng. Belief network powerconstructor. http://www.cs.ualberta.ca/ jcheng/bnpc.htm, 1998.

[2] David Maxwell Chickering. The winmine toolkit. Technical Report MSR-TR-2002-103, Microsoft, Redmond, WA, 2002.

[3] D.E. Edwards. Hierarchical interaction models. *J. Roy. Satist Soc. B*, 52:3–20, 1990.

[4] Alex A. Freitas, editor. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer, 2002.

[5] N. Friedman and I. Nachman. Gaussian procces networks. In *Proceedings of the 16th CUAI*, pages 211–219, 2000.

[6] G.W. Greewood, G.B. Fogel, and M. Ciobanu. Emphasizing extinction in evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 666–671, 1999.

[7] S. Hettich, C.L. Blake, and C.J. Merz. Uci repository of machine learning databases, 1998.

[8] Chrisman L. A roadmap to research on bayesian networks and other decomposable probabilistic models. Technical report, School of Computer Science, 1996.

[9] Wai Lam. Bayesian network refinement via machine learning approach. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 240–251, 1998.

[10] Wai Lam and Fahiem Bacchus. Learning bayesian belief networks an approach based on the mdl principle. In *Computational Intelligence*, pages 269–293, 1994.

[11] J.W. Myers, K.B. Laskey, and T.S. Levitt. Learning bayesian networks fro incomplete data with stochastic search algorithms. In *Proceeding of 15th conference Uncertainty Artificial Intelligence*, pages 476–485, 1999.

[12] P. Myllymaki, T. Silander, H. Tirri, and P. Uronen. B-course: a web service for bayesian data analysis. In *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, pages 247–256, 2001.

[13] R.S. Siegler. Three aspects of cognitive development. In *Cognitive Psychology*, pages 481–520, 1976.

[14] Man-Leung Wong and Kwong-Sak Leung. Evolving program induction directed by logic grammars. In *Evolutionary Computation*, pages 143–180, 1997.

[15] Man Leung Wong and Kwong Sak Leung. An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach. In *IEEE Transactions on Evolutionary Computation*, pages 378–404, 2004.