US 20250265393A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0265393 A1**

BRINTON (43) **Pub. Date: Aug. 21, 2025**

(54) **INFORMATION PROCESSING APPARATUS, INFORMATION PROCESSING METHOD, AND PROGRAM**

(71) Applicant: **NEC Corporation**, Tokyo (JP)

(72) Inventor: **Joe BRINTON**, Tokyo (JP)

(73) Assignee: **NEC Corporation**, Tokyo (JP)

(21) Appl. No.: **19/050,337**

(22) Filed: **Feb. 11, 2025**

(30) **Foreign Application Priority Data**

Feb. 21, 2024 (JP) 2024-024466

Publication Classification

(51) **Int. Cl.**
G06F 30/27 (2020.01)
G06F 30/18 (2020.01)

(52) **U.S. Cl.**
CPC ***G06F 30/27*** (2020.01); ***G06F 30/18*** (2020.01)

(57) **ABSTRACT**

An information processing apparatus of the present disclosure includes: a configuration collecting unit that collects system configuration information included by design path information; a feature value generating unit that generates a feature value of an element in the system configuration information included by the design path information; a distributed representation learning unit that acquires, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements and the feature value of the element; and a converting unit that generates design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

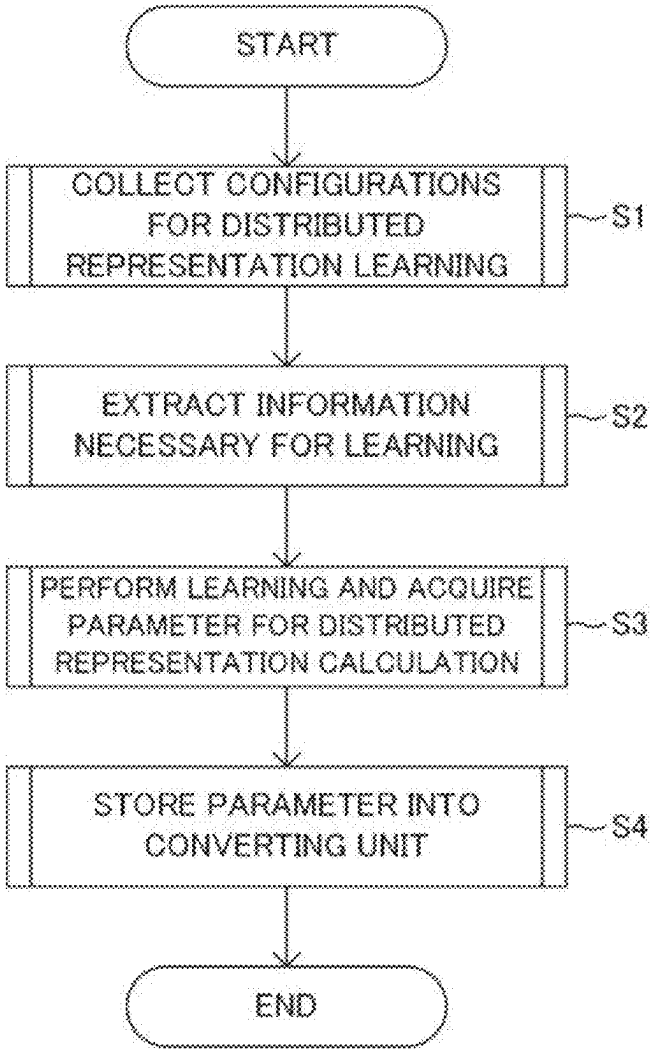


Fig.1

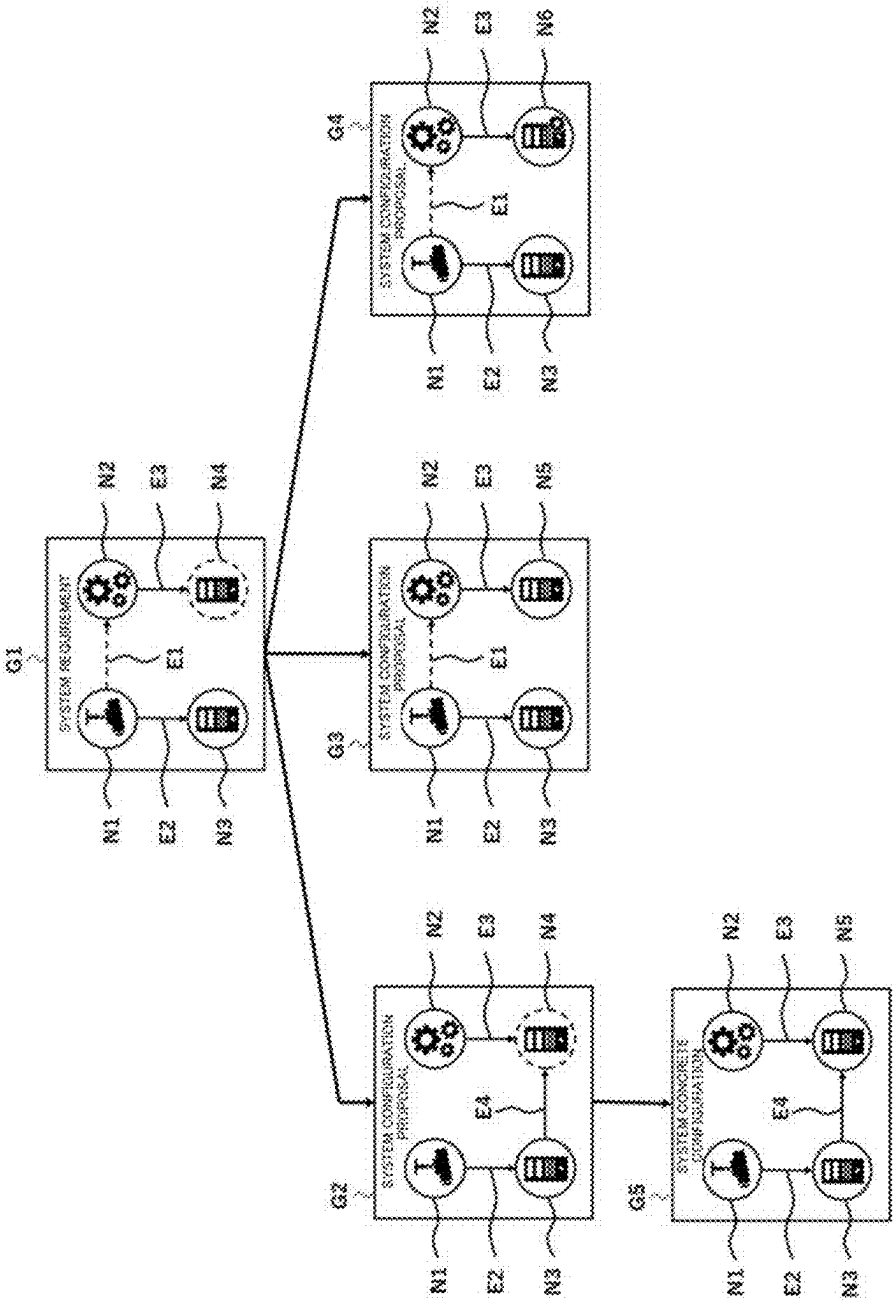

G1
SYSTEM REQUIREMENT
N1
N2
N3
N4
E1
E2
E3
G2
SYSTEM CONFIGURATION PROPOSAL
N1
N2
N3
N4
E1
E2
E3
E4
G3
SYSTEM CONFIGURATION PROPOSAL
N1
N2
N3
N5
E1
E2
E3
G4
SYSTEM CONFIGURATION PROPOSAL
N1
N2
N3
N6
E1
E2
E3
G5
SYSTEM CONFIGURATION
N1
N2
N3
N5
E2
E3

Fig.2

P1

COMPONENT IDENTIFICATION INFORMATION	FUNCTION INFORMATION (TYPE)	CONCRETE/ABSTRACT
N1	CAMERA	1
N2	FACE RECOGNITION SOFTWARE	1
N3	SERVER COMPUTER A	1
N4	SERVER COMPUTER	0
N5	SERVER COMPUTER A	1
N6	SERVER COMPUTER B	1
E1	CONNECT	0
E2	JOIN	1
E3	JOIN	1
E4	HTTP COMMUNICATION	1

Fig.3

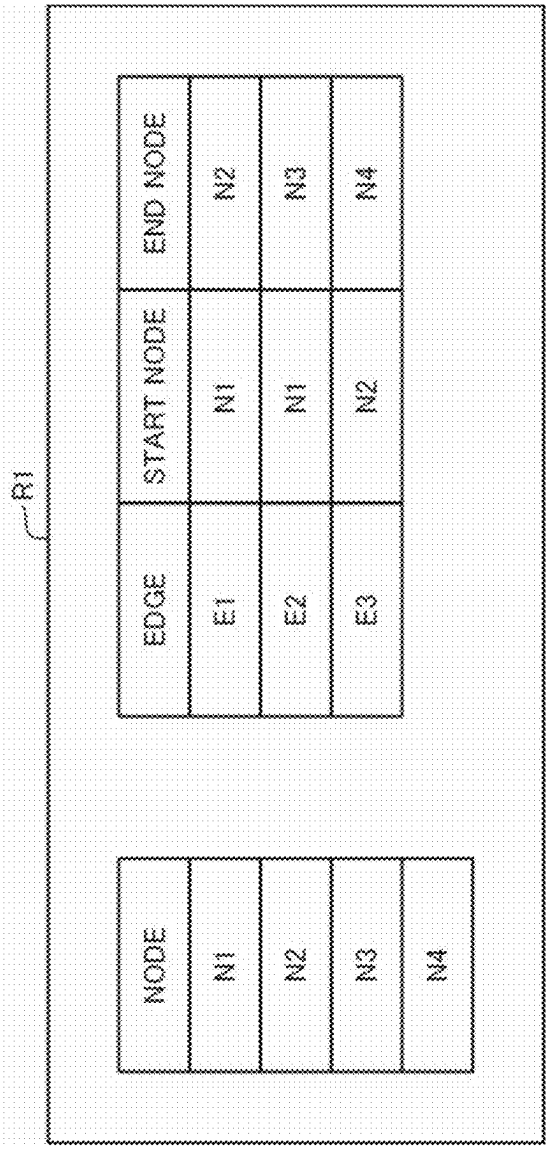

R1
EDGE
START NODE
END NODE
E1
N1
N2
E2
N1
N3
E3
N2
N4
NODE
N1
N2
N3
N4

Fig.4

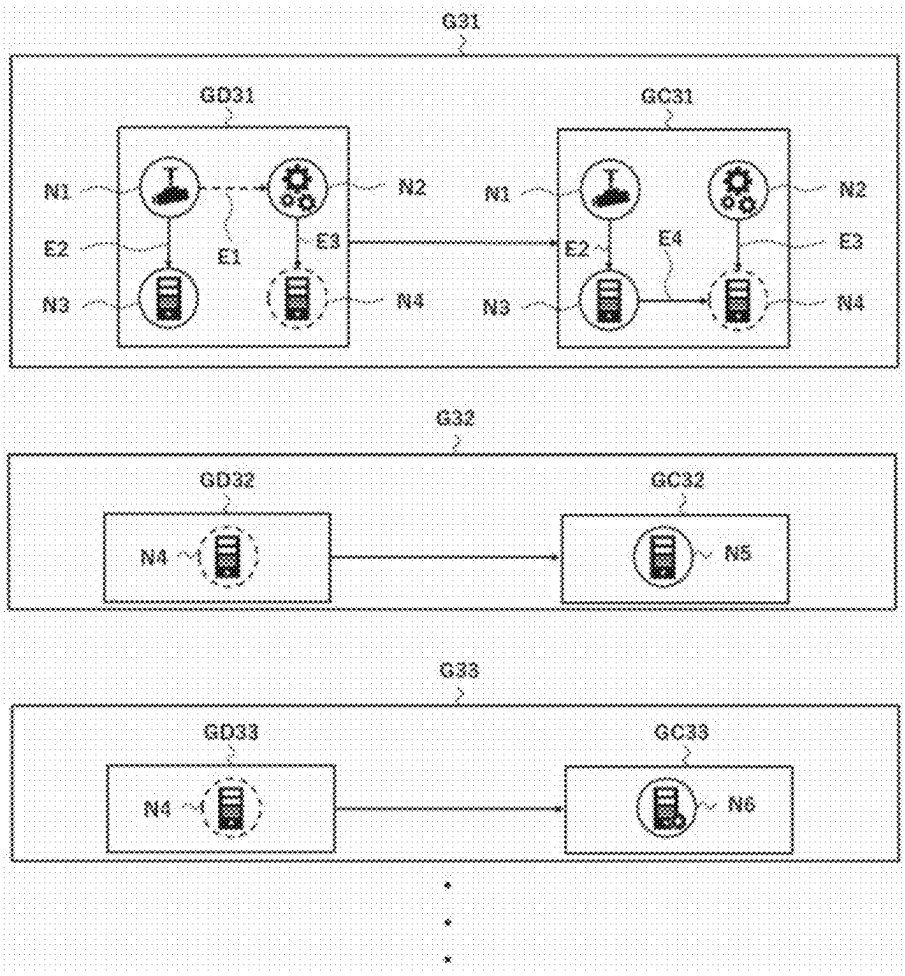

G31
GD31
GC31
N1
N2
E2
E1
E3
N3
N4
N1
N2
E2
E4
E3
N3
N4
G32
GD32
GC32
N4
N5
G33
GD33
GC33
N4
N6

Fig.5

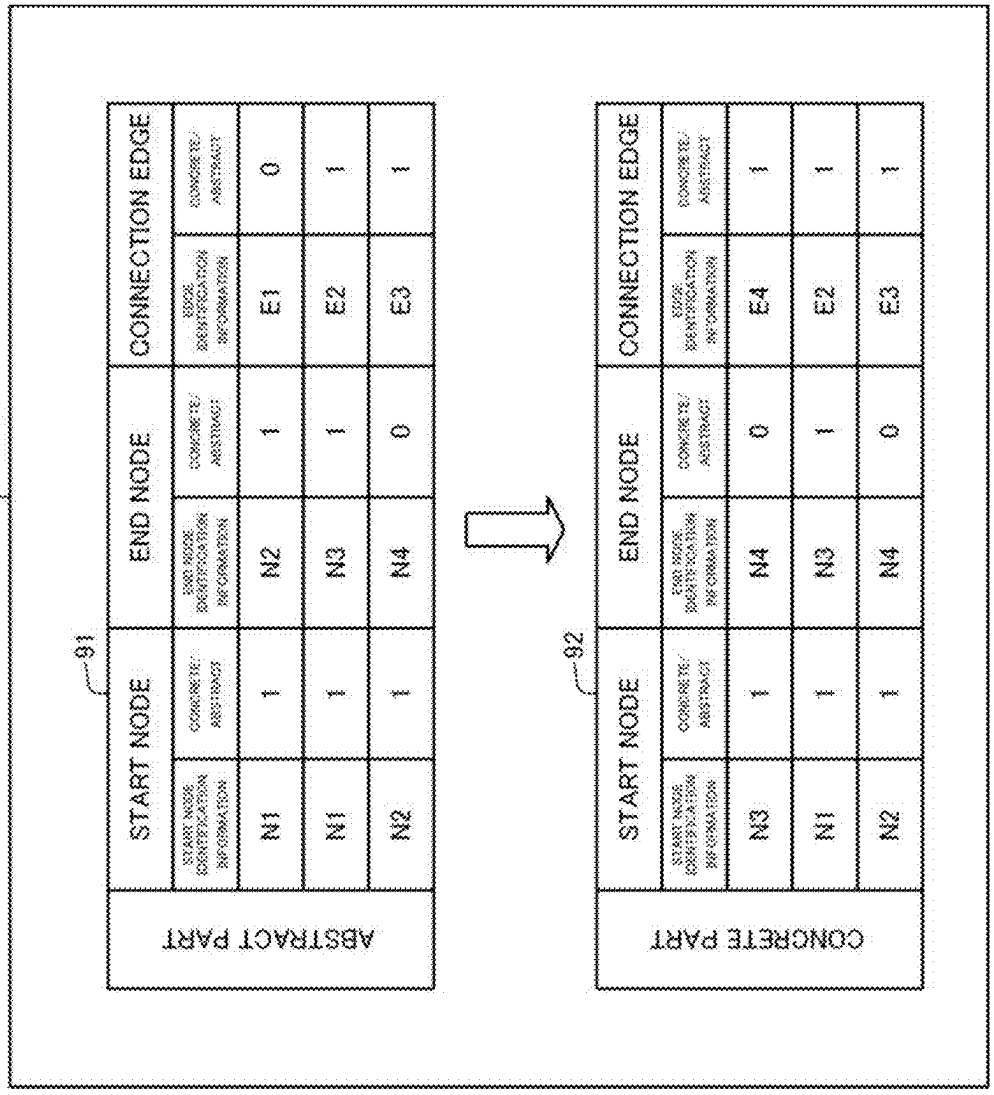

Rule1
91
ABSTRACT PART
START NODE
START NODE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
END NODE
START NODE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
CONNECTION EDGE
EDGE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
N1 1 N2 1 E1 0
N1 1 N3 1 E2 1
N2 1 N4 0 E3 1
92
CONCRETE PART
START NODE
START NODE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
END NODE
START NODE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
CONNECTION EDGE
EDGE IDENTIFICATION INFORMATION
CONCRETE/ ABSTRACT
N3 1 N4 0 E4 1
N1 1 N3 1 E2 1
N2 1 N4 0 E3 1

Fig.6

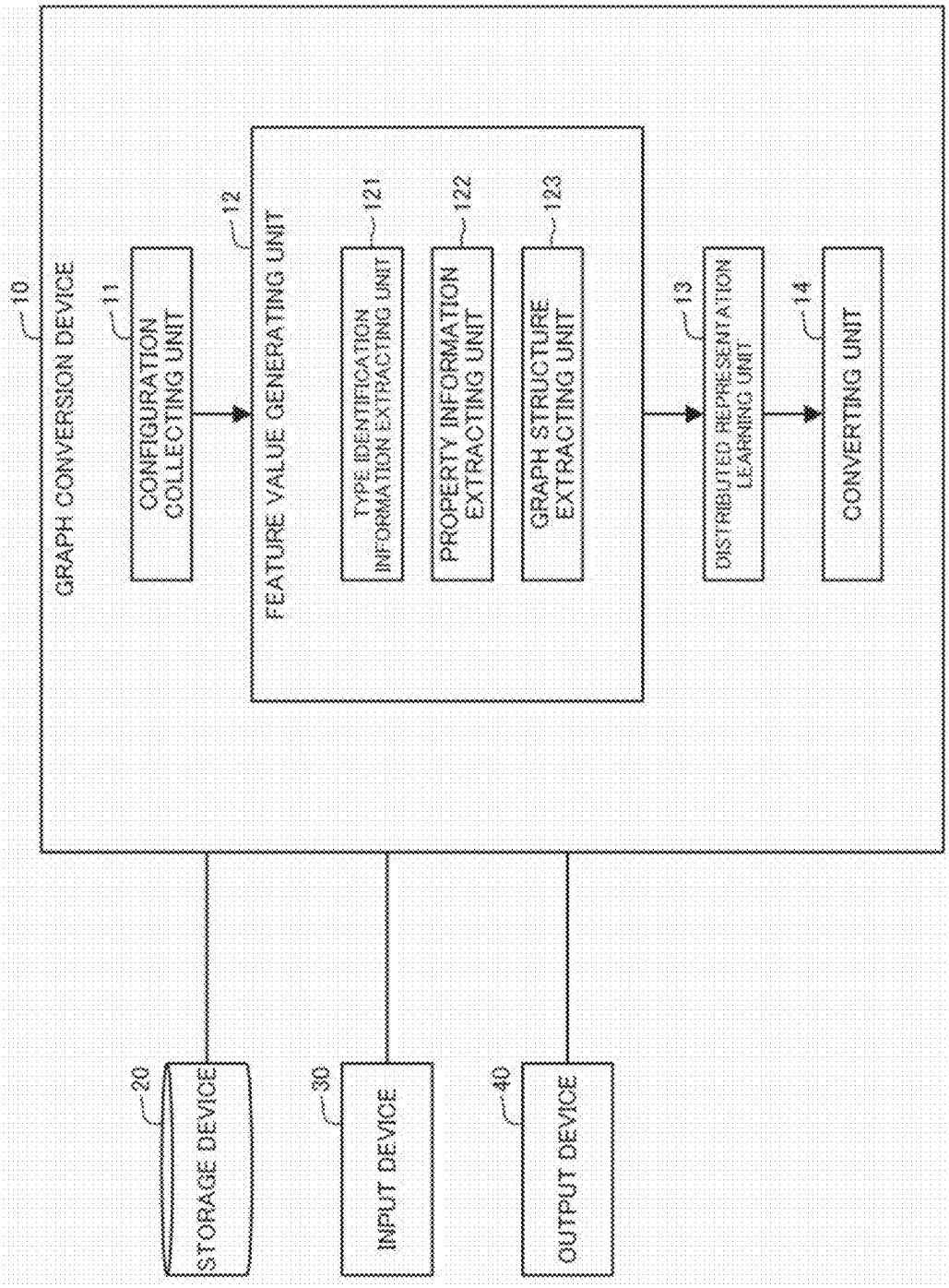

10
GRAPH CONVERSION DEVICE
11
CONFIGURATION COLLECTING UNIT
12
FEATURE VALUE GENERATING UNIT
121
TYPE IDENTIFICATION INFORMATION EXTRACTING UNIT
122
PROPERTY INFORMATION EXTRACTING UNIT
123
GRAPH STRUCTURE EXTRACTING UNIT
13
DISTRIBUTED REPRESENTATION LEARNING UNIT
14
CONVERTING UNIT
20
STORAGE DEVICE
30
INPUT DEVICE
40
OUTPUT DEVICE

Fig. 7

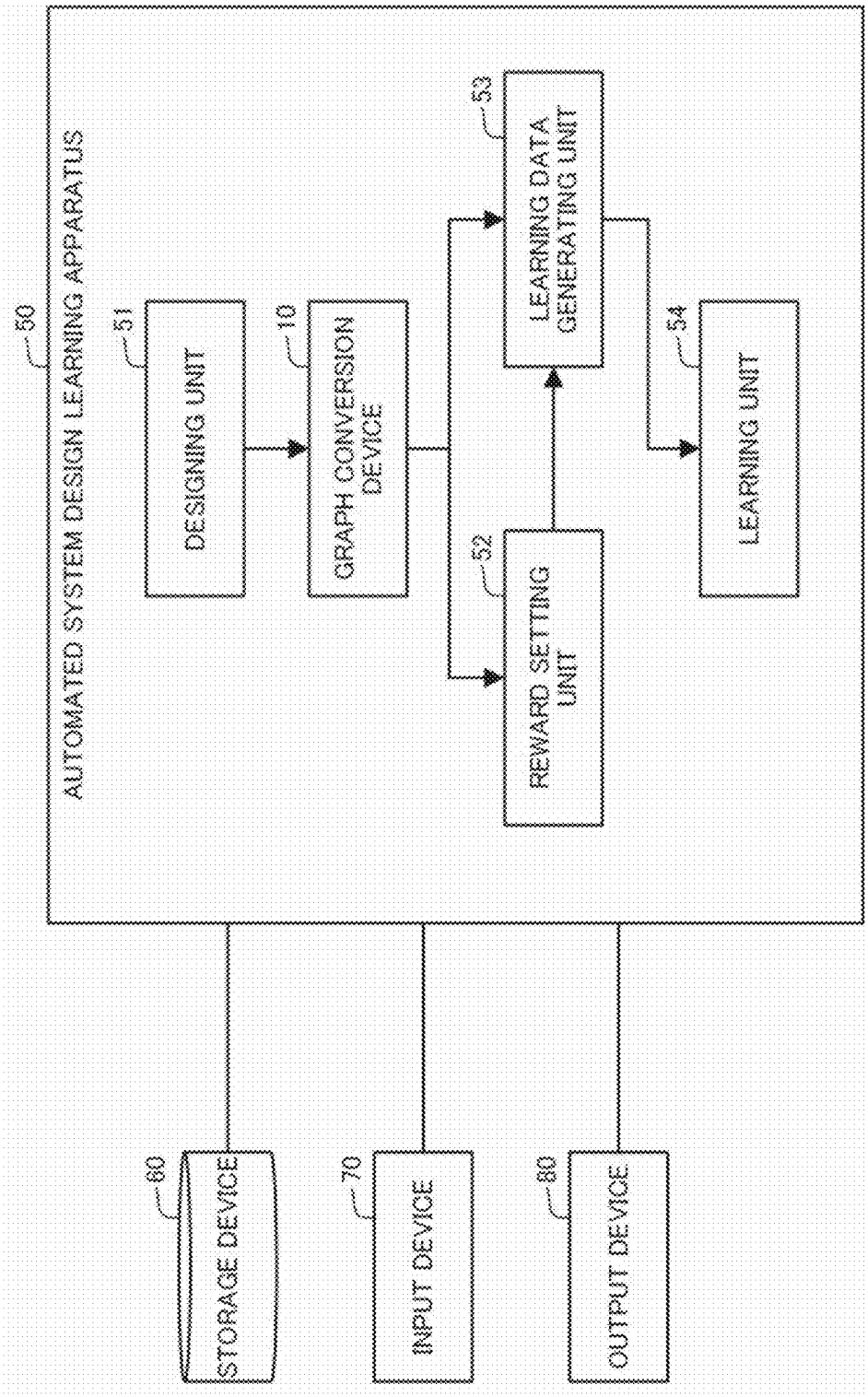

50
AUTOMATED SYSTEM DESIGN LEARNING APPARATUS
51
DESIGNING UNIT
10
GRAPH CONVERSION DEVICE
53
LEARNING DATA GENERATING UNIT
52
REWARD SETTING UNIT
54
LEARNING UNIT
60
STORAGE DEVICE
70
INPUT DEVICE
80
OUTPUT DEVICE

Fig.8

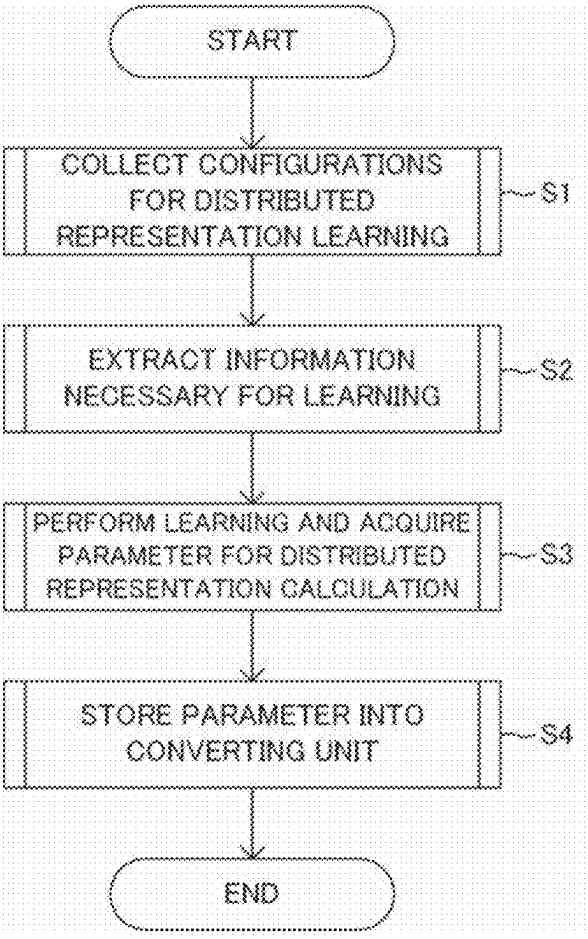

START
COLLECT CONFIGURATIONS FOR DISTRIBUTED REPRESENTATION LEARNING
S1
EXTRACT INFORMATION NECESSARY FOR LEARNING
S2
PERFORM LEARNING AND ACQUIRE PARAMETER FOR DISTRIBUTED REPRESENTATION CALCULATION
S3
STORE PARAMETER INTO CONVERTING UNIT
S4
END

Fig.9

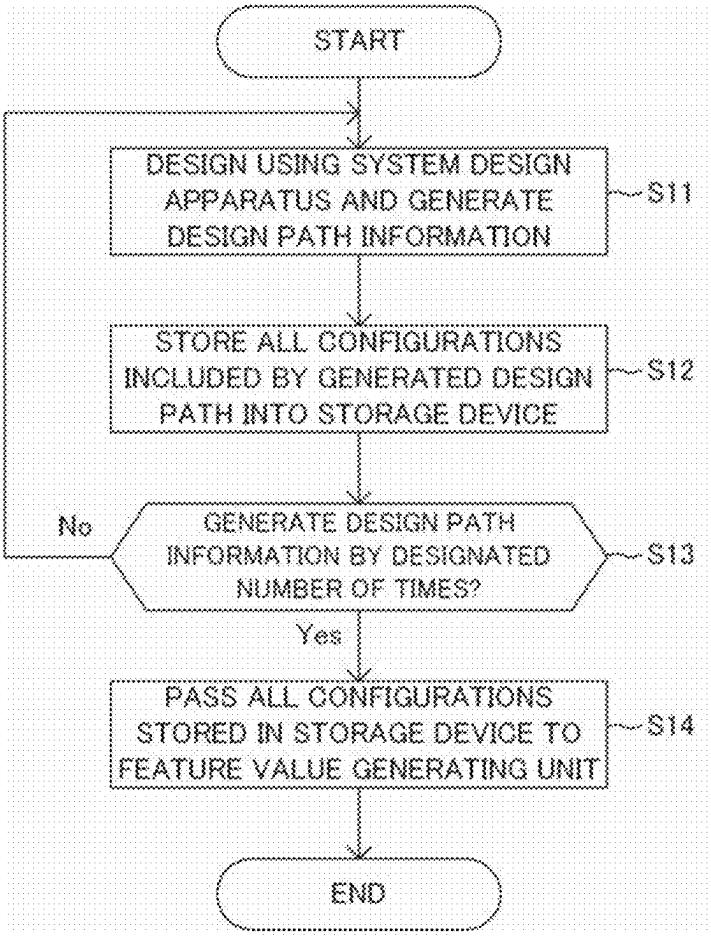

START
DESIGN USING SYSTEM DESIGN APPARATUS AND GENERATE DESIGN PATH INFORMATION
S11
STORE ALL CONFIGURATIONS INCLUDED BY GENERATED DESIGN PATH INTO STORAGE DEVICE
S12
No
GENERATE DESIGN PATH INFORMATION BY DESIGNATED NUMBER OF TIMES?
S13
Yes
PASS ALL CONFIGURATIONS STORED IN STORAGE DEVICE TO FEATURE VALUE GENERATING UNIT
S14
END

Fig.10

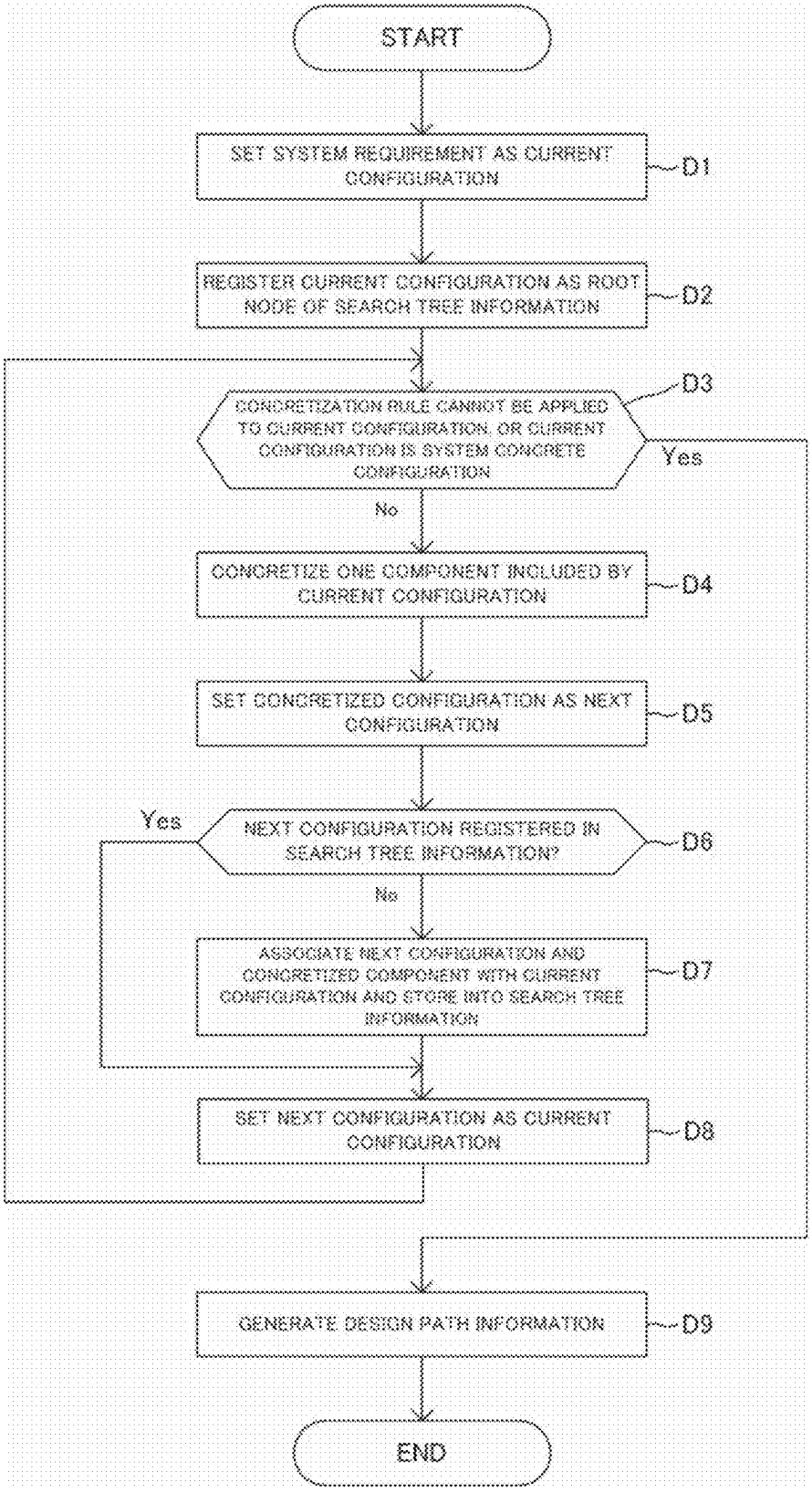

START
SET SYSTEM REQUIREMENT AS CURRENT CONFIGURATION
D1
REGISTER CURRENT CONFIGURATION AS ROOT NODE OF SEARCH TREE INFORMATION
D2
CONCRETIZATION RULE CANNOT BE APPLIED TO CURRENT CONFIGURATION, OR CURRENT CONFIGURATION IS SYSTEM CONCRETE CONFIGURATION
D3
Yes
No
CONCRETIZE ONE COMPONENT INCLUDED BY CURRENT CONFIGURATION
D4
SET CONCRETIZED CONFIGURATION AS NEXT CONFIGURATION
D5
Yes
NEXT CONFIGURATION REGISTERED IN SEARCH TREE INFORMATION?
D6
No
ASSOCIATE NEXT CONFIGURATION AND CONCRETIZED COMPONENT WITH CURRENT CONFIGURATION AND STORE INTO SEARCH TREE INFORMATION
D7
SET NEXT CONFIGURATION AS CURRENT CONFIGURATION
D8
GENERATE DESIGN PATH INFORMATION
D9
END

Fig.11

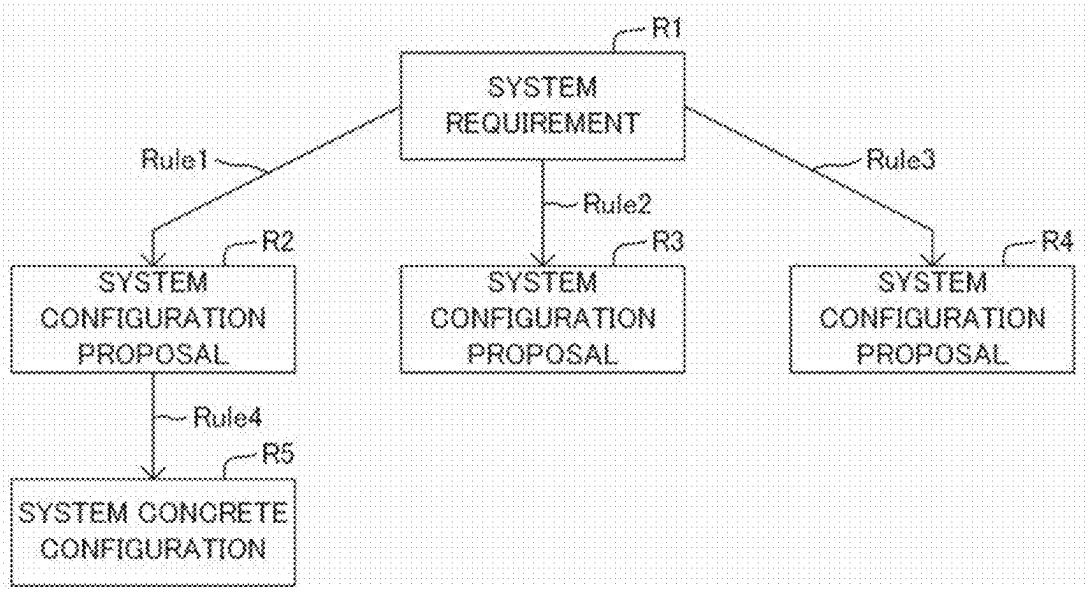

R1
SYSTEM REQUIREMENT
Rule1
Rule2
Rule3
R2
SYSTEM CONFIGURATION PROPOSAL
R3
SYSTEM CONFIGURATION PROPOSAL
R4
SYSTEM CONFIGURATION PROPOSAL
Rule4
R5
SYSTEM CONCRETE CONFIGURATION

Fig.12

T10

ROOT NODE	PARENT NODE	CHILD NODE	DIRECTED EDGE
1	R1	R2	Rule1
1	R1	R3	Rule2
1	R1	R4	Rule3
0	R2	R5	Rule4

Fig.13

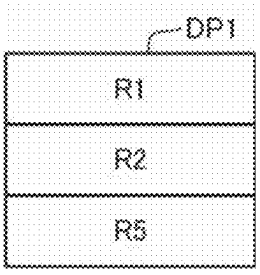


Fig.14

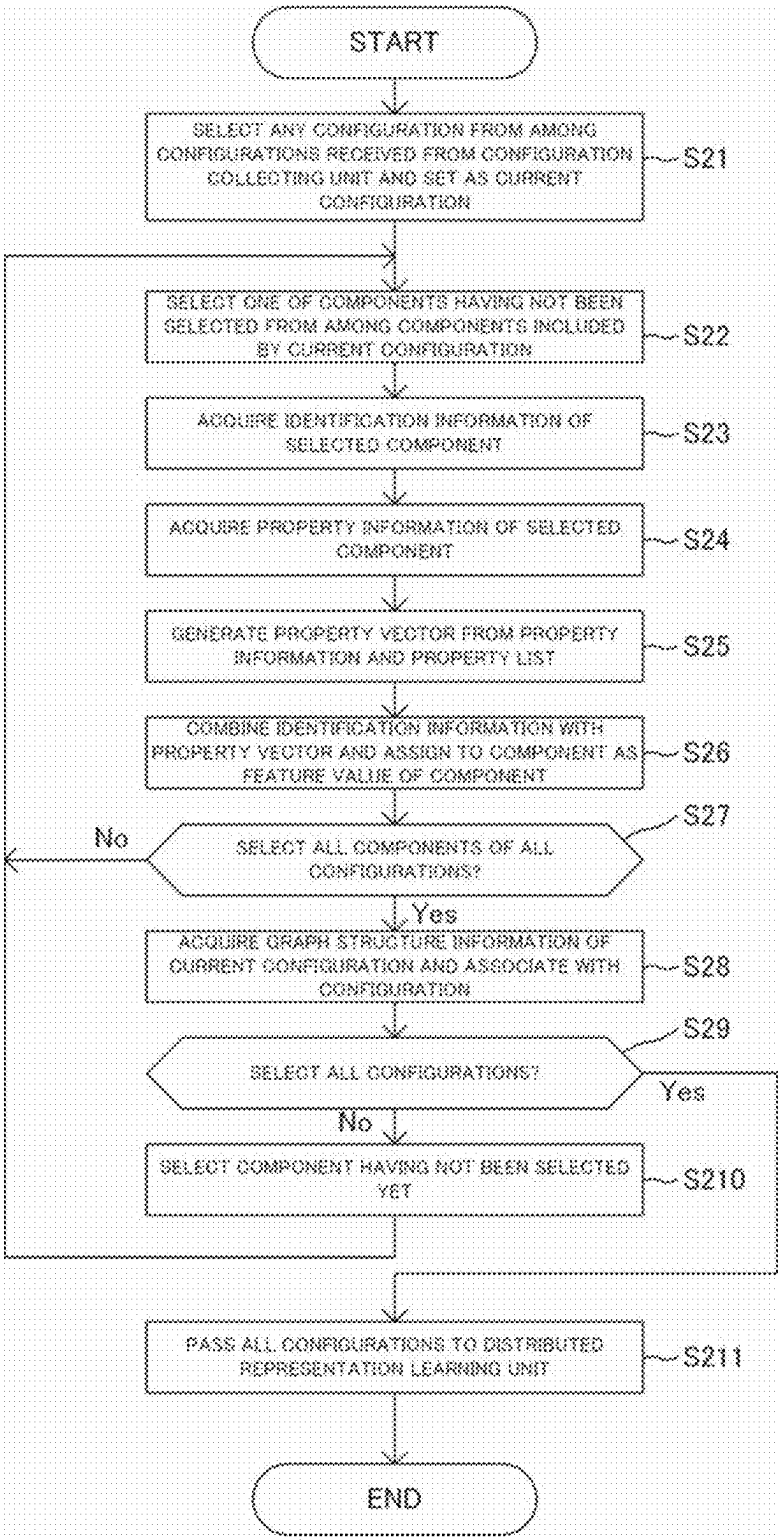

START
SELECT ANY CONFIGURATION FROM AMONG CONFIGURATIONS RECEIVED FROM CONFIGURATION COLLECTING UNIT AND SET AS CURRENT CONFIGURATION
S21
SELECT ONE OF COMPONENTS HAVING NOT BEEN SELECTED FROM AMONG COMPONENTS INCLUDED BY CURRENT CONFIGURATION
S22
ACQUIRE IDENTIFICATION INFORMATION OF SELECTED COMPONENT
S23
ACQUIRE PROPERTY INFORMATION OF SELECTED COMPONENT
S24
GENERATE PROPERTY VECTOR FROM PROPERTY INFORMATION AND PROPERTY LIST
S25
COMBINE IDENTIFICATION INFORMATION WITH PROPERTY VECTOR AND ASSIGN TO COMPONENT AS FEATURE VALUE OF COMPONENT
S26
S27
No
SELECT ALL COMPONENTS OF ALL CONFIGURATIONS?
Yes
ACQUIRE GRAPH STRUCTURE INFORMATION OF CURRENT CONFIGURATION AND ASSOCIATE WITH CONFIGURATION
S28
S29
SELECT ALL CONFIGURATIONS?
Yes
No
SELECT COMPONENT HAVING NOT BEEN SELECTED YET
S210
PASS ALL CONFIGURATIONS TO DISTRIBUTED REPRESENTATION LEARNING UNIT
S211
END

Fig.15

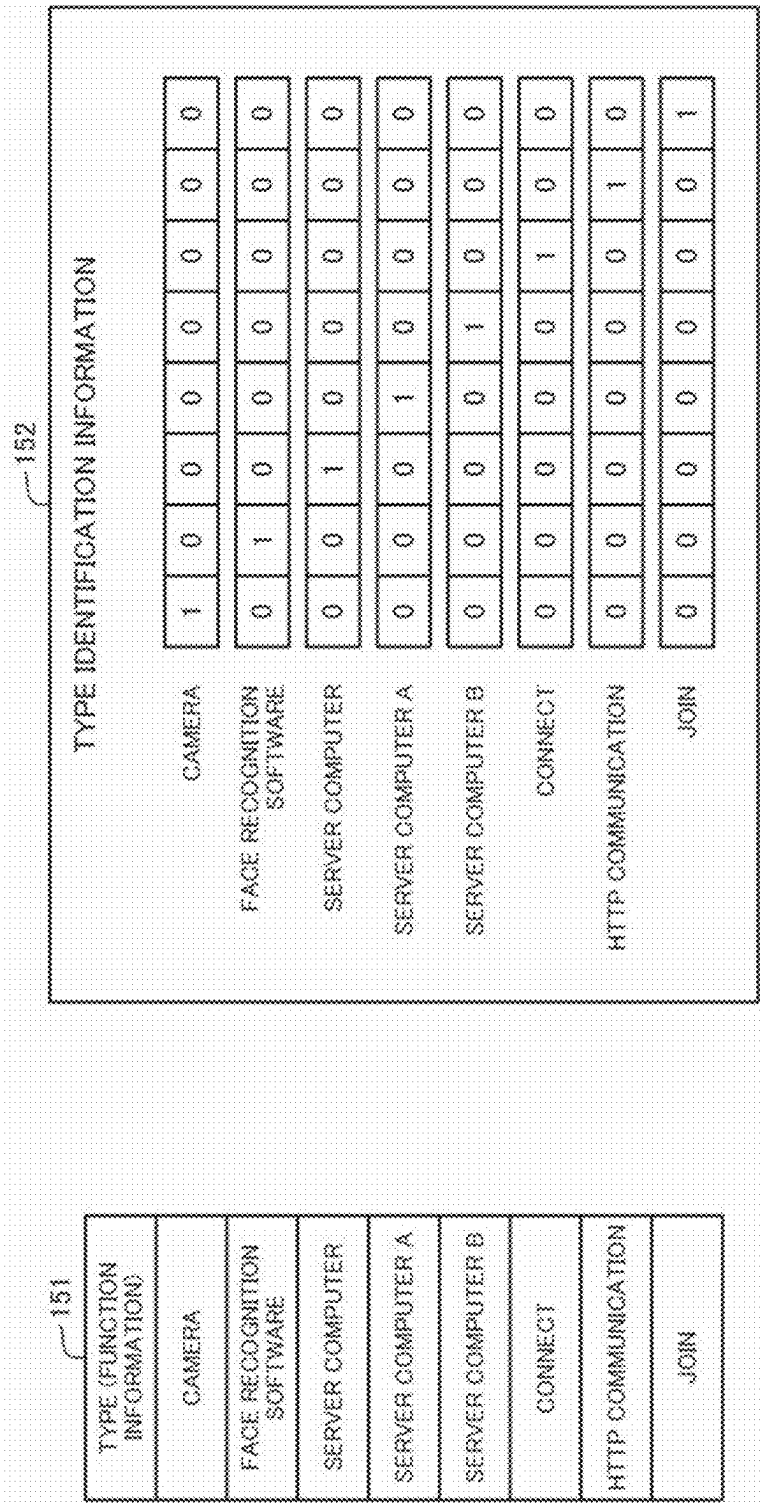

151
TYPE (FUNCTION INFORMATION)
CAMERA
FACE RECOGNITION SOFTWARE
SERVER COMPUTER
SERVER COMPUTER A
SERVER COMPUTER B
CONNECT
HTTP COMMUNICATION
JOIN
152
TYPE IDENTIFICATION INFORMATION
CAMERA 1 0 0 0 0 0 0 0
FACE RECOGNITION SOFTWARE 0 1 0 0 0 0 0 0
SERVER COMPUTER 0 0 1 0 0 0 0 0
SERVER COMPUTER A 0 0 0 1 0 0 0 0
SERVER COMPUTER B 0 0 0 0 1 0 0 0
CONNECT 0 0 0 0 0 1 0 0
HTTP COMMUNICATION 0 0 0 0 0 0 1 0
JOIN 0 0 0 0 0 0 0 1

Fig.16

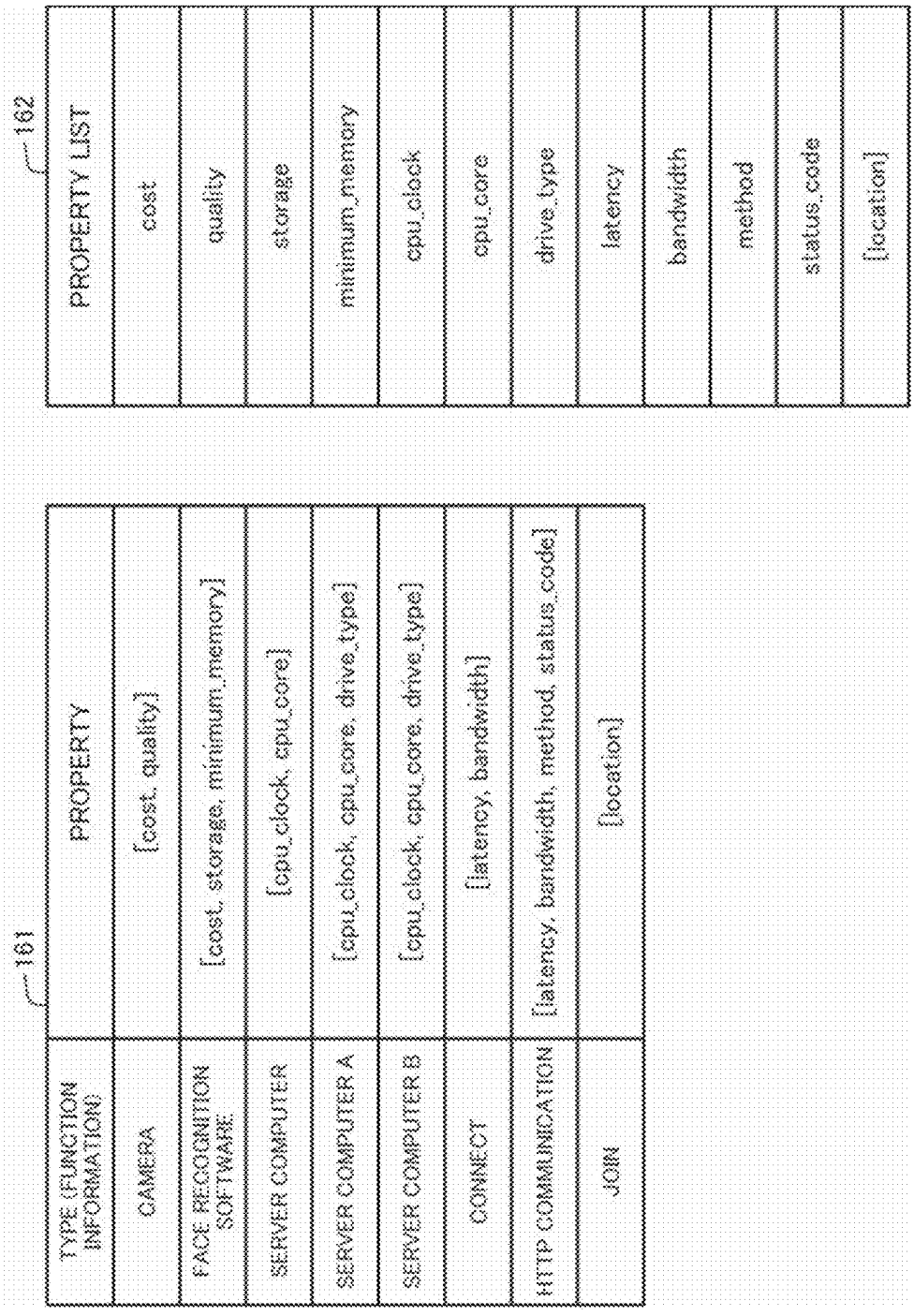

161
TYPE (FUNCTION INFORMATION)
PROPERTY
CAMERA
[cost, quality]
FACE RECOGNITION SOFTWARE
[cost, storage, minimum_memory]
SERVER COMPUTER
[cpu_clock, cpu_core]
SERVER COMPUTER A
[cpu_clock, cpu_core, drive_type]
SERVER COMPUTER B
[cpu_clock, cpu_core, drive_type]
CONNECT
[latency, bandwidth]
HTTP COMMUNICATION
[latency, bandwidth, method, status_code]
JOIN
[location]
162
PROPERTY LIST
cost
quality
storage
minimum_memory
cpu_clock
cpu_core
drive_type
latency
bandwidth
method
status_code
[location]

Fig.17

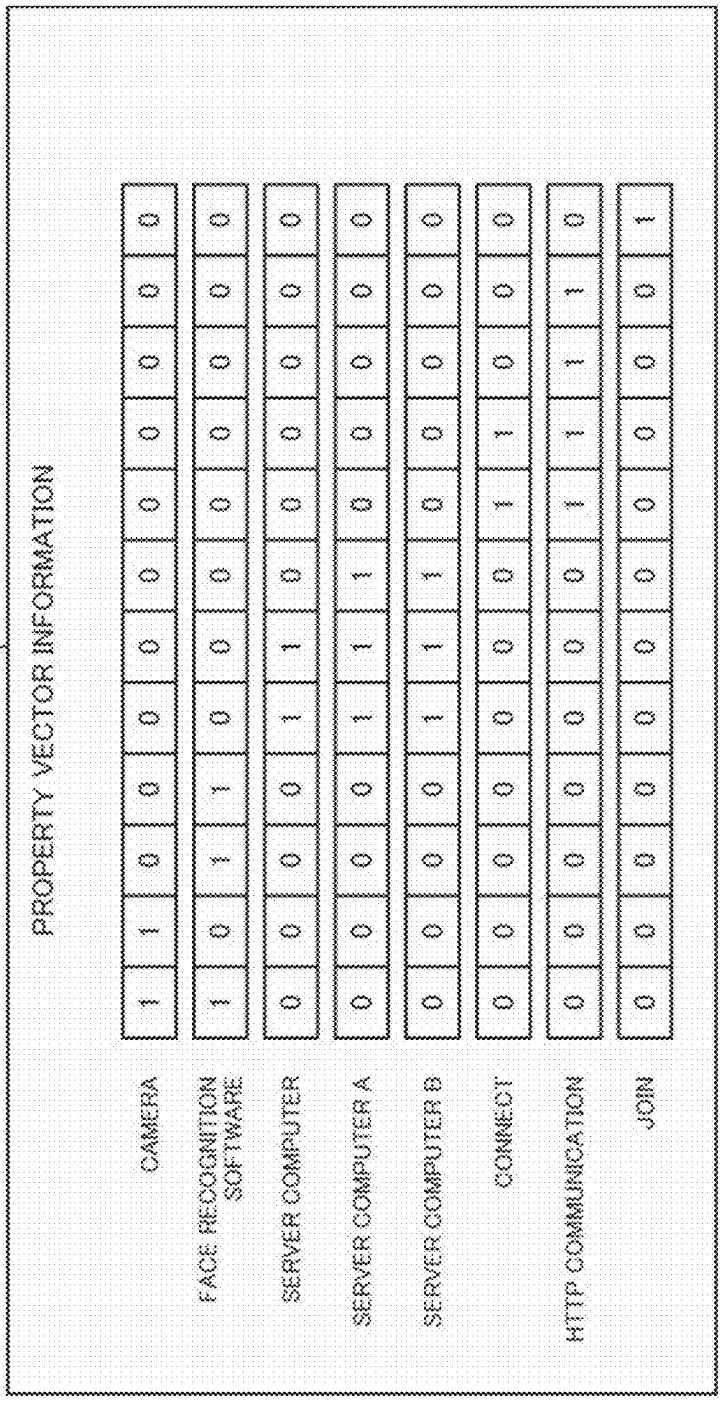

171
PROPERTY VECTOR INFORMATION
CAMERA
FACE RECOGNITION SOFTWARE
SERVER COMPUTER
SERVER COMPUTER A
SERVER COMPUTER B
CONNECT
HTTP COMMUNICATION
JOIN

Fig.18

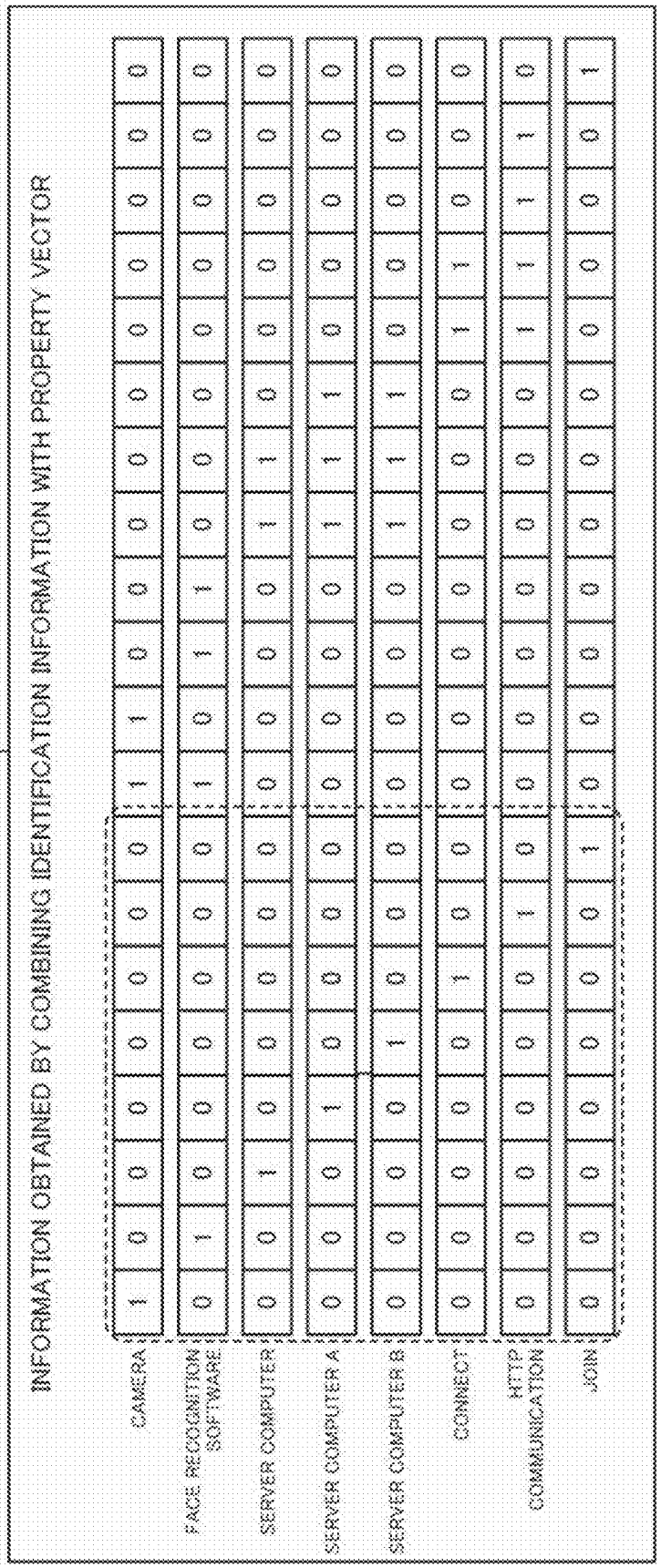

181
INFORMATION OBTAINED BY COMBINING IDENTIFICATION INFORMATION WITH PROPERTY VECTOR
CAMERA
FACE RECOGNITION SOFTWARE
SERVER COMPUTER
SERVER COMPUTER A
SERVER COMPUTER B
CONNECT
HTTP COMMUNICATION
JOIN

Fig.19

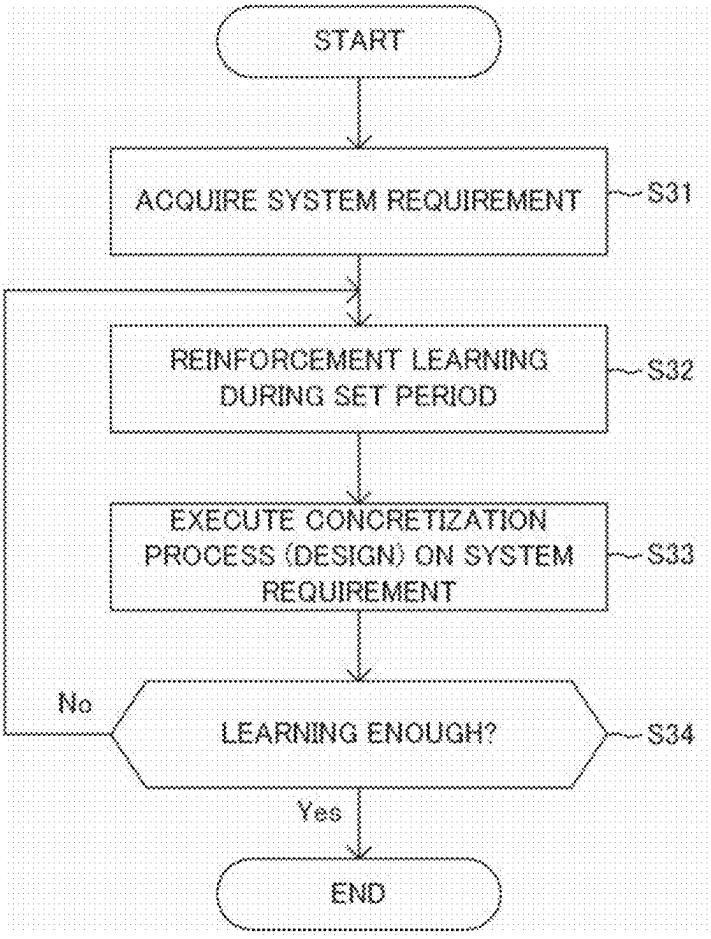

START
ACQUIRE SYSTEM REQUIREMENT
S31
REINFORCEMENT LEARNING DURING SET PERIOD
S32
EXECUTE CONCRETIZATION PROCESS (DESIGN) ON SYSTEM REQUIREMENT
S33
No
LEARNING ENOUGH?
S34
Yes
END

Fig.20

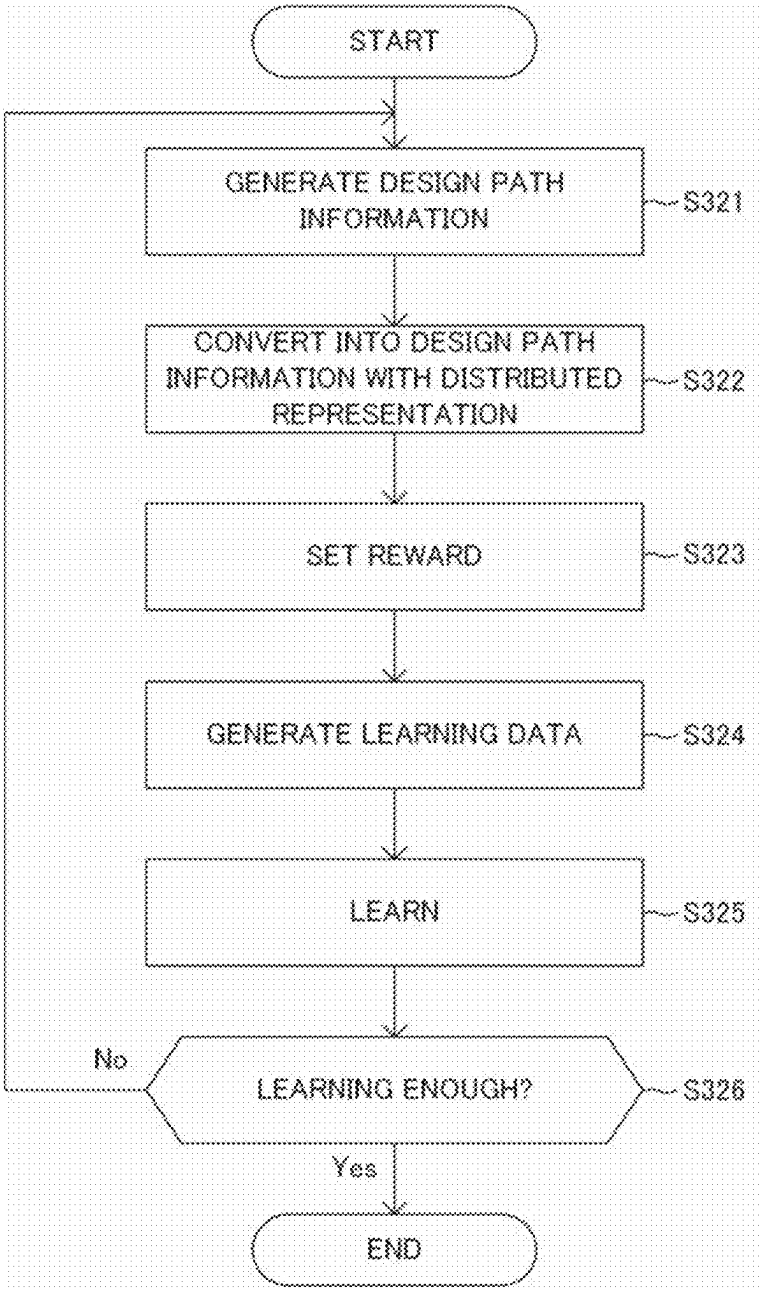

START
GENERATE DESIGN PATH INFORMATION
S321
CONVERT INTO DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION
S322
SET REWARD
S323
GENERATE LEARNING DATA
S324
LEARN
S325
No
LEARNING ENOUGH?
S326
Yes
END

Fig.21

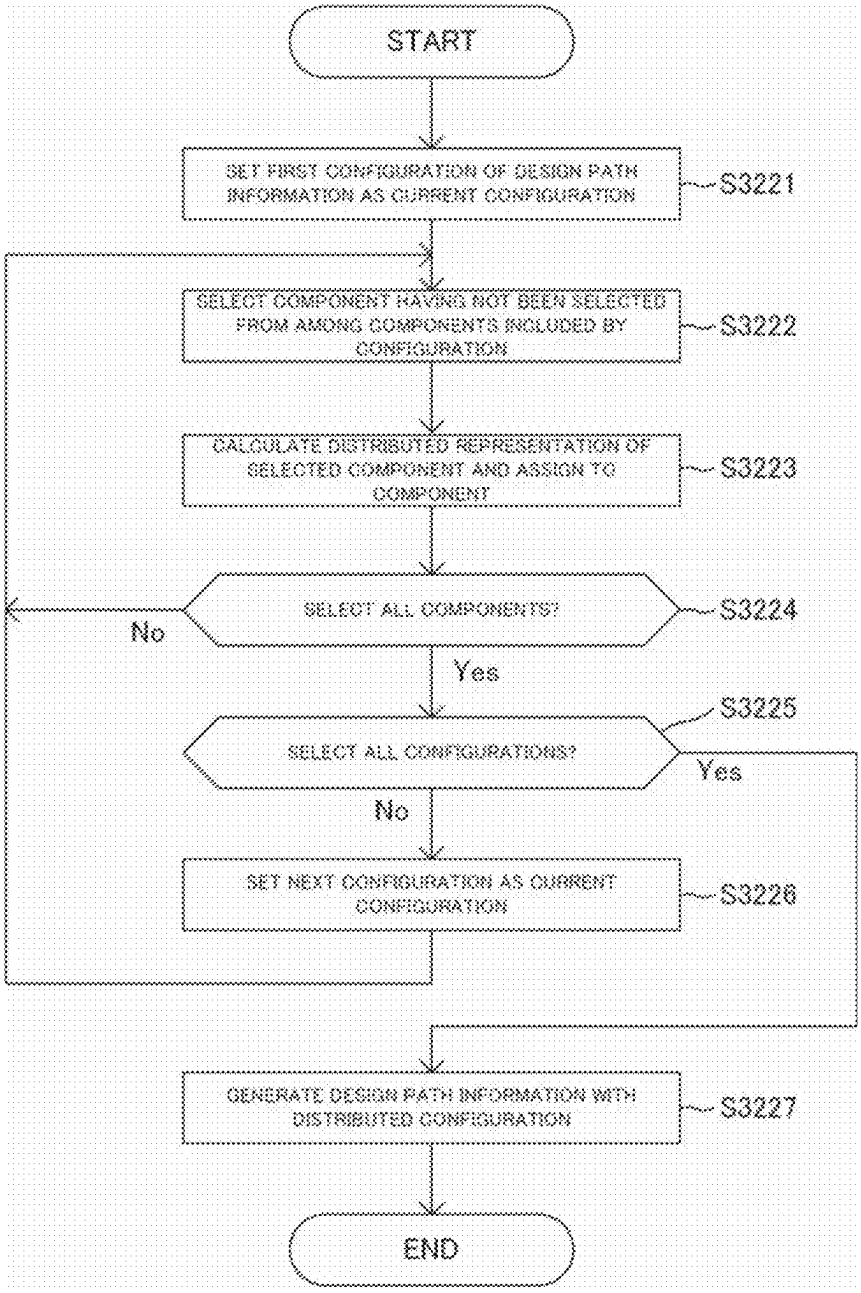

START
SET FIRST CONFIGURATION OF DESIGN PATH INFORMATION AS CURRENT CONFIGURATION
S3221
SELECT COMPONENT HAVING NOT BEEN SELECTED FROM AMONG COMPONENTS INCLUDED BY CONFIGURATION
S3222
CALCULATE DISTRIBUTED REPRESENTATION OF SELECTED COMPONENT AND ASSIGN TO COMPONENT
S3223
SELECT ALL COMPONENTS?
S3224
No
Yes
S3225
SELECT ALL CONFIGURATIONS?
Yes
No
SET NEXT CONFIGURATION AS CURRENT CONFIGURATION
S3226
GENERATE DESIGN PATH INFORMATION WITH DISTRIBUTED CONFIGURATION
S3227
END

Fig.22

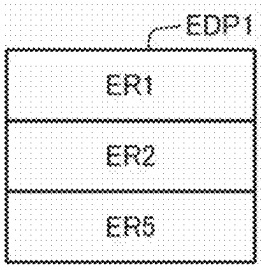


Fig.23

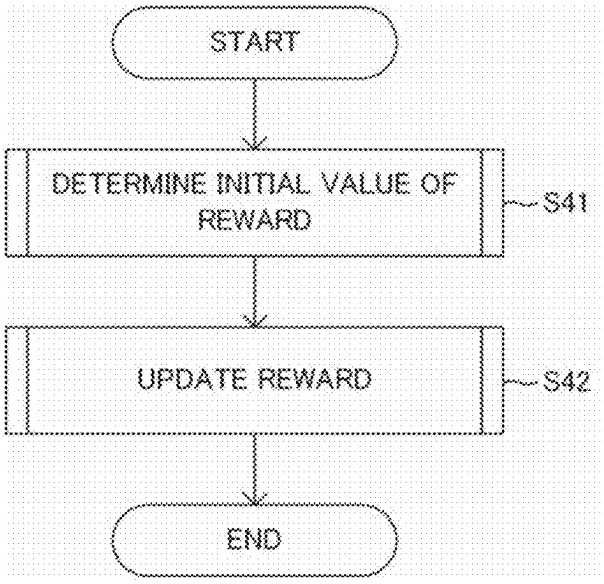

START
DETERMINE INITIAL VALUE OF REWARD
S41
UPDATE REWARD
S42
END

Fig.24

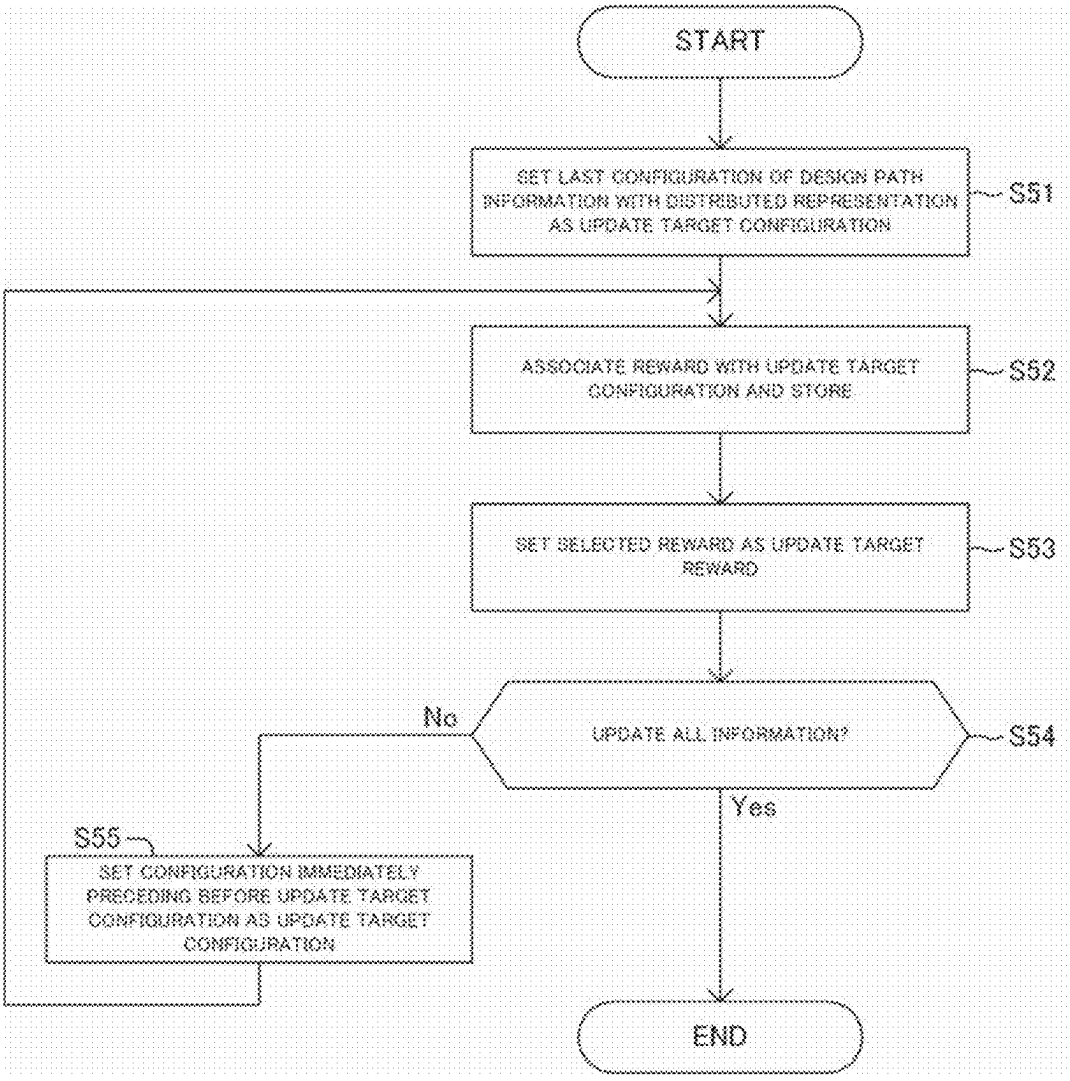

START
SET LAST CONFIGURATION OF DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION AS UPDATE TARGET CONFIGURATION
S51
ASSOCIATE REWARD WITH UPDATE TARGET CONFIGURATION AND STORE
S52
SET SELECTED REWARD AS UPDATE TARGET REWARD
S53
UPDATE ALL INFORMATION?
S54
No
Yes
S55
SET CONFIGURATION IMMEDIATELY PRECEDING BEFORE UPDATE TARGET CONFIGURATION AS UPDATE TARGET CONFIGURATION
END

Fig.25

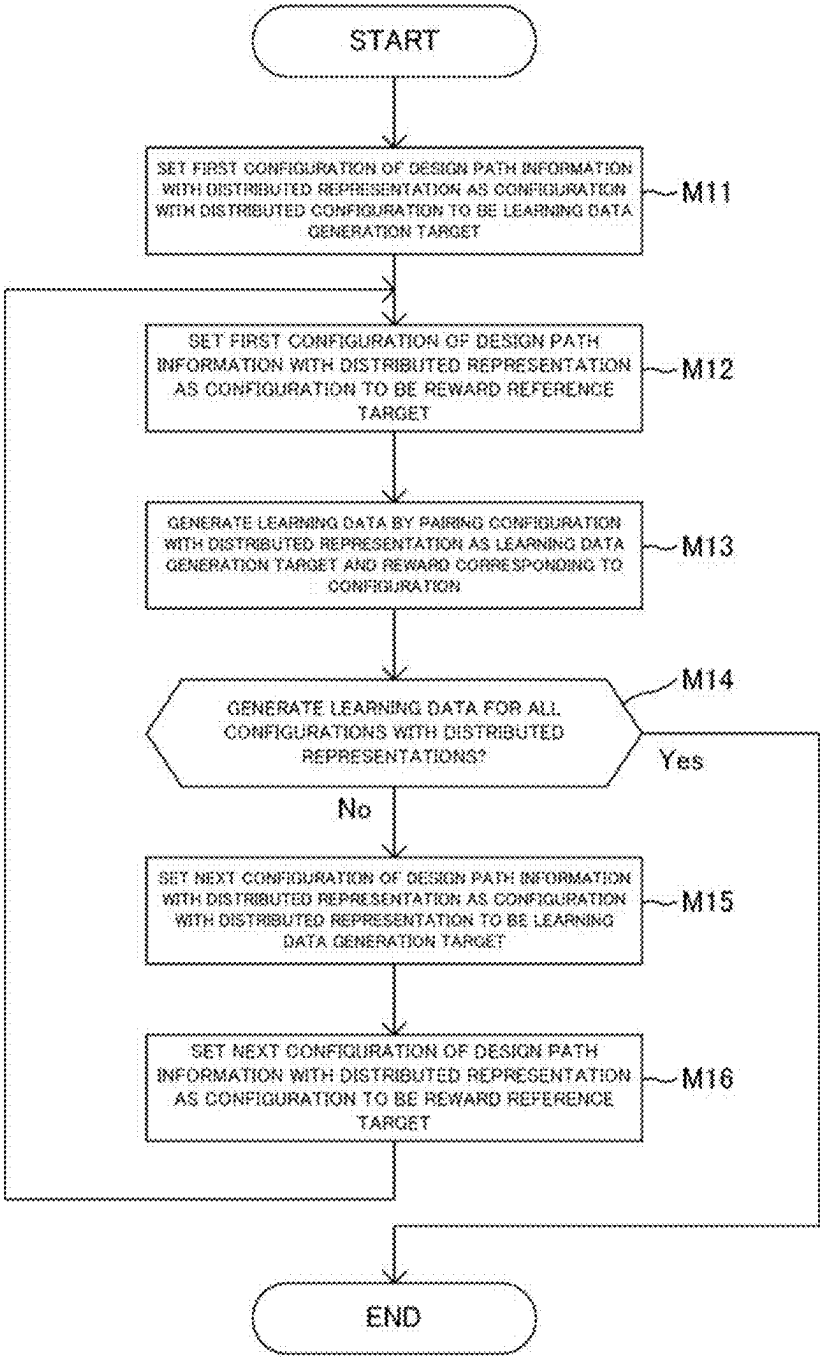

START
SET FIRST CONFIGURATION OF DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION AS CONFIGURATION WITH DISTRIBUTED CONFIGURATION TO BE LEARNING DATA GENERATION TARGET
M11
SET FIRST CONFIGURATION OF DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION AS CONFIGURATION TO BE REWARD REFERENCE TARGET
M12
GENERATE LEARNING DATA BY PAIRING CONFIGURATION WITH DISTRIBUTED REPRESENTATION AS LEARNING DATA GENERATION TARGET AND REWARD CORRESPONDING TO CONFIGURATION
M13
GENERATE LEARNING DATA FOR ALL CONFIGURATIONS WITH DISTRIBUTED REPRESENTATIONS?
M14
Yes
No
SET NEXT CONFIGURATION OF DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION AS CONFIGURATION WITH DISTRIBUTED REPRESENTATION TO BE LEARNING DATA GENERATION TARGET
M15
SET NEXT CONFIGURATION OF DESIGN PATH INFORMATION WITH DISTRIBUTED REPRESENTATION AS CONFIGURATION TO BE REWARD REFERENCE TARGET
M16
END

Fig.26

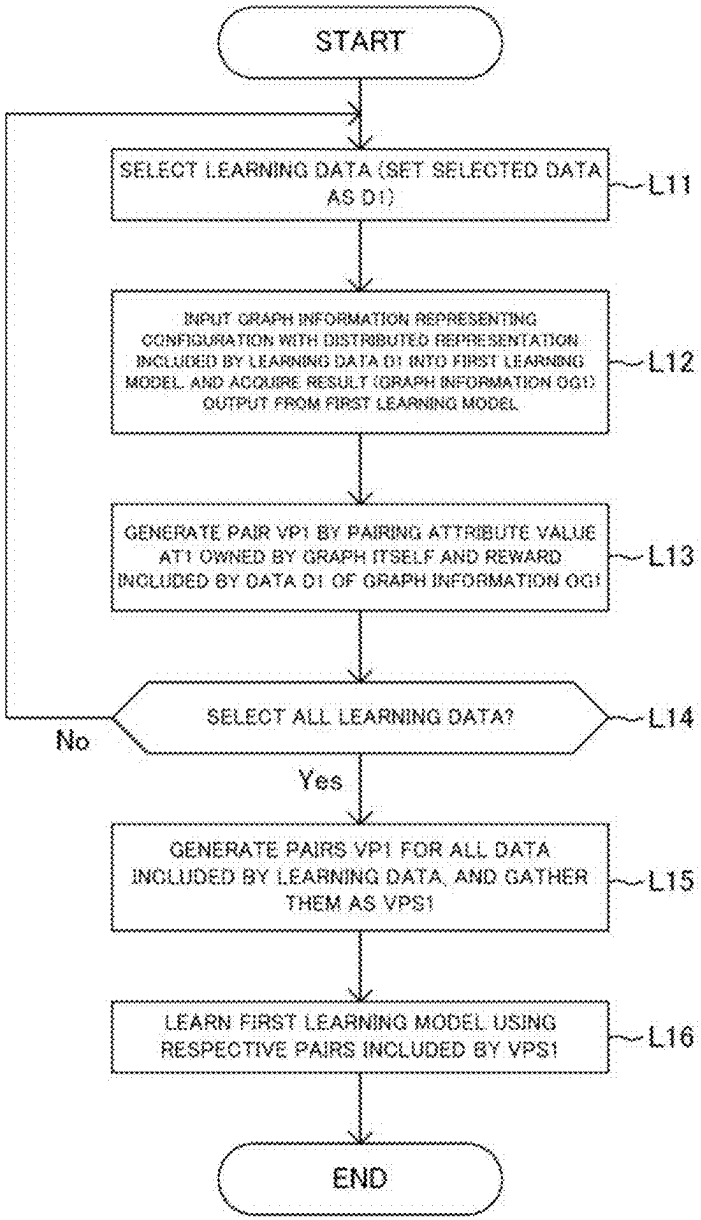

START
SELECT LEARNING DATA (SET SELECTED DATA AS D1)
L11
INPUT GRAPH INFORMATION REPRESENTING CONFIGURATION WITH DISTRIBUTED REPRESENTATION INCLUDED BY LEARNING DATA D1 INTO FIRST LEARNING MODEL, AND ACQUIRE RESULT (GRAPH INFORMATION OG1) OUTPUT FROM FIRST LEARNING MODEL
L12
GENERATE PAIR VP1 BY PAIRING ATTRIBUTE VALUE AT1 OWNED BY GRAPH ITSELF AND REWARD INCLUDED BY DATA D1 OF GRAPH INFORMATION OG1
L13
SELECT ALL LEARNING DATA?
L14
No
Yes
GENERATE PAIRS VP1 FOR ALL DATA INCLUDED BY LEARNING DATA, AND GATHER THEM AS VPS1
L15
LEARN FIRST LEARNING MODEL USING RESPECTIVE PAIRS INCLUDED BY VPS1
L16
END

Fig.27

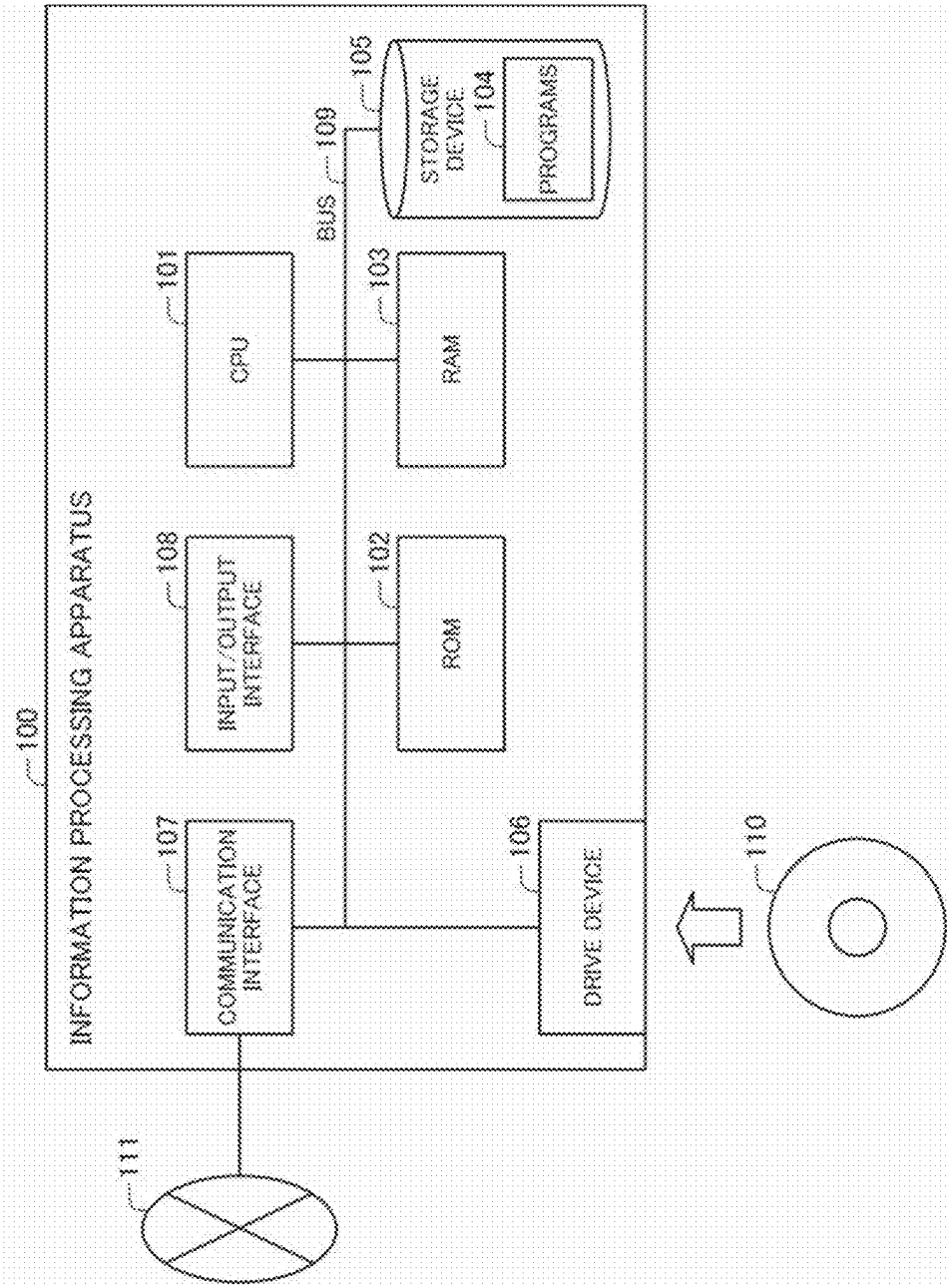
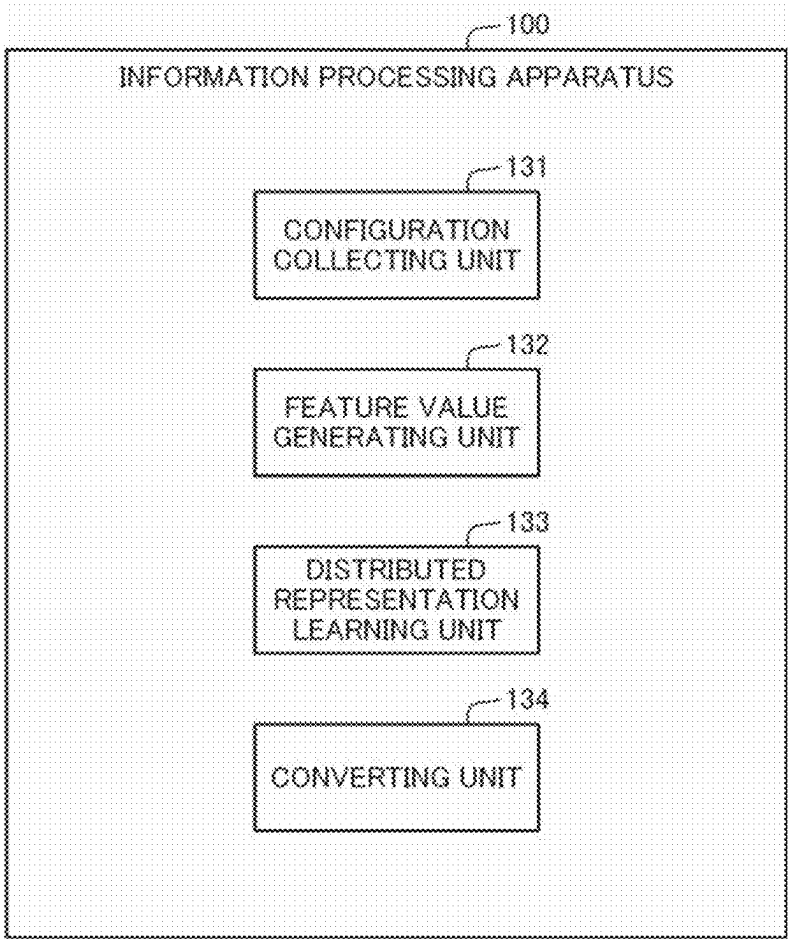

100
INFORMATION PROCESSING APPARATUS
101
CPU
103
RAM
108
INPUT/OUTPUT INTERFACE
102
ROM
107
COMMUNICATION INTERFACE
106
DRIVE DEVICE
BUS
109
105
STORAGE DEVICE
104
PROGRAMS
110
111

Fig.28


100
INFORMATION PROCESSING APPARATUS
131
CONFIGURATION COLLECTING UNIT
132
FEATURE VALUE GENERATING UNIT
133
DISTRIBUTED REPRESENTATION LEARNING UNIT
134
CONVERTING UNIT

INFORMATION PROCESSING APPARATUS, INFORMATION PROCESSING METHOD, AND PROGRAM

INCORPORATION BY REFERENCE

[0001] This application is based upon and claims the benefit of priority from Japanese patent application No. 2024-024466, filed on Feb. 21, 2024, the disclosure of which is incorporated herein in its entirety by reference.

TECHNICAL FIELD

[0002] The present disclosure relates to an information processing apparatus, an information processing method, and a program.

BACKGROUND ART

[0003] When designing an ICT (Information and Communication Technology) system, first, in requirements definition, the designer creates system requirements, which are information summarizing customer's requirements. At this time, system requirements generally include abstract information and often do not represent a deployable system configuration as they are. Therefore, the designer concretizes stepwise the abstract information included in the system requirements and designs a deployable system configuration in which all the information are concrete finally.

[0004] Since this series of actions requires expertise in elements included in a system configuration and the relation between the elements, man-hours related to design and necessary human costs increase as a system becomes sophisticated.

[0005] As a method for solving the above problem, an automated system design technique has been proposed. The automated system design technique is a technique of automatically deriving a system configuration including only concrete information based on system requirements including abstract information.

[0006] As a related technique, Patent Literature 1 discloses a system configuration derivation apparatus that can concretize stepwise abstract parts included in system requirements based on the requirements and derive a deployable and concrete ICT system configuration.

[0007] The system configuration derivation apparatus of Patent Literature 1 learns a method for calculating a score representing the degree of adequacy of a concretization rule that is a rule for rewriting in more detail part of configuration requirements representing requirements concerning an ICT system.

[0008] When an abstract system requirement is entered, the apparatus having thus learned can select adequate concretization rules and design a concrete system configuration speedily and automatically.

[0009] Patent Literature 1: Japanese Patent No. 6989014

[0010] Non-Patent Literature 1: Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. Neural Information Processing Systems.

[0011] Non-patent Literature 2: Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio', P., & Bengio, Y. (2017). Graph Attention Networks. ArXiv, abs/1710.10903.

[0012] However, the technique described in the Patent Literature 1 above does not guarantee accuracy in precision of evaluation by the apparatus for a new element (unlearned element) that does not exist in the past learning process. Therefore, when designing a system using an unlearned element, the apparatus cannot utilize a learning experience concerning a previously learned element for evaluation of the quality of the unlearned element, and cannot perform highly precise system design.

[0013] Therefore, the technique described in Patent Literature 1 has a problem that the apparatus needs relearning when it is desired to design a system using a new element, which requires time cost.

SUMMARY OF THE INVENTION

[0014] Accordingly, an object of the present disclosure is to solve the abovementioned problem that time cost is required for system design.

[0015] An information processing apparatus as an aspect of the present disclosure includes: a configuration collecting unit that collects system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule; a feature value generating unit that generates a feature value of the element in the system configuration information included by the design path information; a distributed representation learning unit that acquires, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and a converting unit that, for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generates design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

[0016] Further, an information processing method as an aspect of the present disclosure is a method by an information processing apparatus, and the method includes: collecting system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule; generating a feature value of the element in the system configuration information included by the design path information; acquiring, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generating design path information with

distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

[0017] Further, a program as an aspect of the present disclosure includes instructions for causing an information processing apparatus to execute processes to: collect system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule; generate a feature value of the element in the system configuration information included by the design path information; acquire, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generate design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

[0018] Configured as described above, the present disclosure can reduce time cost when performing system design.

BRIEF DESCRIPTION OF DRAWINGS

[0019] FIG. **1** is a diagram for describing automatic design of a system in the present disclosure;

[0020] FIG. **2** is a diagram for describing automatic design of the system in the present disclosure;

[0021] FIG. **3** is a diagram for describing automatic design of the system in the present disclosure;

[0022] FIG. **4** is a diagram for describing automatic design of the system in the present disclosure;

[0023] FIG. **5** is a diagram for describing automatic design of the system in the present disclosure;

[0024] FIG. **6** is a block diagram showing the configuration of a graph conversion device in the present disclosure;

[0025] FIG. **7** is a block diagram showing the configuration of an automated system design learning apparatus in the present disclosure;

[0026] FIG. **8** is a flowchart showing an example of the operation of the graph conversion device in the present disclosure;

[0027] FIG. **9** is a flowchart showing an example of operation of collecting a system configuration in the present disclosure;

[0028] FIG. **10** is a flowchart showing an example of operation of generating design path information in the present disclosure;

[0029] FIG. **11** is a diagram showing an example of a search tree in the present disclosure;

[0030] FIG. **12** is a diagram showing an example of a data structure of search tree information in the present disclosure;

[0031] FIG. **13** is a diagram showing an example of a data structure of the design path information in a first example embodiment of the present disclosure;

[0032] FIG. **14** is a flowchart showing an example of operation of extracting information necessary for learning in the present disclosure;

[0033] FIG. **15** is a diagram showing an example of a data structure of type definition information, and a type and identification information corresponding to the type in the present disclosure;

[0034] FIG. **16** is a diagram showing an example of a type and a property defined by the type in the present disclosure;

[0035] FIG. **17** is a diagram showing an example of a property vector generated from type property information and a property list in the present disclosure;

[0036] FIG. **18** is a diagram showing information generated by combining type identification information and the property vector in the present disclosure;

[0037] FIG. **19** is a flowchart showing an example of operation of an automated system design learning apparatus in the present disclosure;

[0038] FIG. **20** is a flowchart showing an example of an overall flow of operation of reinforcement learning of the automated system design learning apparatus in the present disclosure;

[0039] FIG. **21** is a flowchart showing an example of operation in which the graph conversion device in the present disclosure generates design path information with distributed representation in the present disclosure;

[0040] FIG. **22** is a diagram showing an example of a data structure of the design path information with distributed representation in the present disclosure;

[0041] FIG. **23** is a flowchart showing an example of operation of setting a reward in the present disclosure;

[0042] FIG. **24** is a flowchart showing an example of operation of updating a reward in the present disclosure;

[0043] FIG. **25** is a flowchart showing an example of operation of generating learning data for learning by an automated system design learning apparatus in the present disclosure;

[0044] FIG. **26** is a flowchart showing an example of operation of learning of a learning model in the present disclosure;

[0045] FIG. **27** is a block diagram showing a hardware configuration of an information processing apparatus in the present disclosure; and

[0046] FIG. **28** is a block diagram showing a configuration of the information processing apparatus in the present disclosure.

EXAMPLE EMBODIMENT

First Example Embodiment

[0047] A first example embodiment of the present disclosure will be described with reference to the drawings. The drawings may be relevant to any of the example embodiments.

[Overview of Existing Technique]

[0048] To begin with, the overview of automatic design of an ICT system in this example embodiment will be described. First, the premise is that the requirement and configuration of an ICT system can be expressed with a

graph. A graph represents elements (components and the relation between the components) included in the configuration of an ICT system using nodes or edges. A node is an element that represents, for example, a device or an application. An edge is an element that represents a connection relation between two nodes.

[0049] As an example of an ICT system, a case of designing a face recognition system will be described. When designing a face recognition system, the designer first creates a system requirement for the face recognition system.

[0050] FIG. **1** is a diagram for describing automatic design of a face recognition system. A graph G**1** in FIG. **1** is a graph showing the configuration of a system requirement R**1** of the face recognition system. The graph G**1** in FIG. **1** is shown using nodes N**1**, N**2**, N**3** and N**4** and edges E**1**, E**2** and E**3**.

[0051] The node N**1** represents a camera function as a concrete element (solid-line circle), the node N**2** represents a face recognition function as a concrete element (solid-line circle), the node N**3** represents a server computer function as a concrete element (solid-line circle), and the node N**4** represents a server computer function as an abstract element (dashed circle). The edge E**1** represents an HTTP communication function as an abstract element (dashed arrow), and the edges E**2** and E**3** represent join functions as concrete elements (solid-line arrows). The parts other than the graph G**1** (system requirement) in FIG. **1** will be described later.

[0052] FIG. **2** is a diagram for explaining the data structure of component information that is information of the above-mentioned elements. Component information P**1** shown in FIG. **2** associates, for each of the component identification information for identifying nodes and edges, function information (also referred to as type) of the node or edge with information showing whether the node or edge is a concrete component (element) or an abstract component (element) (concrete (1)/abstract (0)).

[0053] FIG. **3** is a diagram for explaining the data structure of a system requirement. The system requirement R**1** of FIG. **3** is a system requirement corresponding to the graph G**1** shown in FIG. **1**. The system requirement R**1** includes component identification information for identifying all the nodes included in the requirement. Then, the system requirement R**1** is information including component identification information for identifying all the edges included in the requirement, component identification information for identifying the node equivalent to the start of the edge, and component identification information for identifying the node equivalent to the end of the edge.

[0054] In FIGS. **1** to **3** shown above, a graph, component information, and a system requirement are described using the graph G**1**, the component information P**1**, and the system requirement R**1**, but a graph, component information, and a system requirement are not limited to the graph G**1**, the component information P**1**, and the system requirement R**1** shown in FIGS. **1** to **3**.

[0055] Next, conversion from an abstract part into a concrete part in a graph in automated system design will be described. The example of FIG. **1** shows that graphs G**2**, G**3** and G**4** shown in FIG. **1** are derived by concretizing the graph G**1** corresponding to the system requirement R**1** based on a concretization rule to be described later. The graphs G**2**, G**3** and G**4** are system configuration proposals. A system configuration proposal includes an abstract component and is not a system requirement. As in the above, it is shown that a system concrete configuration G**5** including no abstract element is derived by concretizing the system configuration proposal G**2** using a concretization rule.

[0056] Further, although not illustrated in the example of FIG. **1**, there may be a system configuration proposal derived by concretizing the graph G**1** corresponding to the system requirement R**1**, other than the graphs G**2**, G**3** and G**4** of FIG. **1**.

[0057] Next, a concretization rule will be described. A concretization rule is a rule for converting an abstract part included in a system requirement or a system configuration proposal into a concrete part. A concretization rule is information including detection information corresponding to an abstract part of a graph and conversion information corresponding to a concrete part of the graph.

[0058] Detection information and conversion information are graphs each including a node, an edge, or both, and one conversion information is associated with one detection information.

[0059] When the whole or part of a graph corresponding to a system requirement or a system configuration proposal matches the detection information included by a certain concretization rule, it is possible to concretize by applying the concretization rule and replacing the matching part with the conversion information. An example of the concretization rule will be described later with reference to FIG. **4**. An example of the data structure of the concretization rule will be described later with reference to FIG. **5**.

[0060] Although not illustrated in the example of FIG. **3**, there are also system configuration proposals R**2**, R**3** and R**4** corresponding to the graphs G**2**, G**3** and G**4**. Like the system requirement R**1**, each of the system configuration proposals R**2**, R**3** and R**4** is information including component identification information of all the nodes included in the configuration proposal, component identification information of all the edges, component identification information of the start nodes of the edges, and component identification information of the end nodes of the edges.

[0061] FIG. **4** is a diagram for explaining a concretization rule. FIG. **4** shows graphs G**31**, G**32**, G**33**, . . . , which are graphs representing a plurality of concretization rules.

[0062] The graph G**31** is a graph representing a concretization rule Rule**1** used for converting the graph G**1** shown in FIG. **1** into the graph G**2** shown in FIG. **1** (delete the edge E**1**, add an edge E**4**). A graph GD **31** included by the graph G**31** is a graph representing detection information, and a graph GC**31** is a graph representing conversion information. Since the graph GD **31** matches the graph G**1** shown in FIG. **1**, the concretization rule Rule**1** can be applied. Then, by applying the concretization rule Rule**1**, the configuration of the graph G**1** is replaced with the configuration of the graph GC**31** and becomes the graph G**2**.

[0063] Further, the graph G**32** is a graph representing a concretization rule Rule**2** used for converting the graph G**1** shown in FIG. **1** into the graph G**3** shown in FIG. **1** (convert the node N**4** into a node N**5**). The graph G**33** is a graph representing a concretization rule Rule**3** used for converting the graph G**1** shown in FIG. **1** into the graph G**4** shown in FIG. **1** (convert the node N**4** into a node N**6**). As in the graph G**31**, a graph GD**32** included by the graph G**32** is a graph representing detection information, and a graph GC**32** is a graph representing conversion information. Since the graph GD **32** matches part of the graph G**1** shown in FIG. **1**, the concretization rule Rule**2** can be applied. Then, by applying the concretization rule Rule**2**, part of the graph G**1** (i.e., the

node N4) is replaced with the graph GC32 (i.e., the node N5), and the graph G1 is thereby converted into the graph G3. Moreover, as in the graph G31, a graph GD33 included by the graph G33 is a graph representing detection information, and a graph GC33 is a graph representing conversion information. Since the graph GD 33 matches part of the graph G1 shown in FIG. 1, the concretization rule Rule3 can be applied. Then, by applying the concretization rule Rule3, part of the graph G1 (i.e., the node N4) is replaced with the graph GC33 (i.e., the node N6), and the graph G1 is thereby converted into the graph G4.

[0064] FIG. 5 is a diagram for explaining the data structure of a concretization rule. The concretization rule Rule1 shown in FIG. 5 is the concretization rule corresponding to the graph G31. The concretization rule Rule1 includes detection information 91 used for detecting an abstract part and conversion information 92 for converting the abstract part detected using the detection information 91 into a concrete part. Although not illustrated, there are the concretization rules Rule2, Rule3, . . . like that shown in FIG. 5 corresponding to the graphs G32, G33, . . . for the graphs G32, G33, ...

[0065] For deriving the graphs G2, G3 and G4 from the graph G1 shown in FIG. 1, first, the system requirement R1 is compared with a plurality of detection information (correspond to abstract part in concretization rule), and detection information that matches an abstract part included in the system requirement R1 is detected.

[0066] Next, using the detected detection information corresponding to the abstract part, the abstract part detected from the system requirement R1 is converted into conversion information (concrete part in concretization rule). The conversion into the graph G2 is done by replacement with the conversion information 92 of the concretization rule Rule1 because the system requirement R1 matches the detection information 91 of the concretization rule Rule1. As a result, the edge E1 is deleted from the system requirement R1 and the edge E4 is added.

[0067] Although not illustrated here, the system configuration proposal R2 corresponding to the graph G2 is generated. The system configuration proposals R3 and R4 corresponding to the graphs G3 and G4, respectively, are also generated.

[0068] For each of the plurality of system configuration proposals generated in this manner (system configuration proposals R2, R3, and R4), an abstract part in the system configuration proposal is converted into a concrete part using the concretization rule. In a case where a system configuration proposal as an abstract system configuration is further generated as a result of converting the abstract part in the system configuration proposal into a concrete part using the concretization rule, the abovementioned concretization process is repeated. Then, when a system configuration proposal with no abstract part is generated, the concretization process is stopped (automatic design is ended).

[0069] Here, in the case of concretizing a system requirement and a system configuration proposal, a plurality of concretization rules are used, so that a system configuration proposal and a system concrete configuration to be derived vary in accordance with a selected concretization rule and the order of selection of concretization rules. That is to say, system configuration proposals and system concrete configurations with different configurations are derived in accordance with the types of concretization rules to be selected and the order thereof.

[0070] Further, when the number of concretization rules to be applied increases, the number of system configuration proposals and system concrete configurations to be generated becomes enormous. Furthermore, the plurality of different system concrete configurations may include a system concrete configuration that does not satisfy a system requirement. Therefore, as the number of concretization rules to be applied increases, it becomes difficult to efficiently derive a system concrete configuration.

[0071] Therefore, in order to efficiently derive a system concrete configuration, it is important to appropriately determine the quality of a concretization rule to be applied.

[0072] Then, a learning model is used to select a concretization rule to be applied. That is to say, evaluation is performed on system configuration proposals generated by execution of the concretization process, using a learning model acquired by machine learning, and a system configuration proposal with the highest evaluation value is selected from among the system configuration proposals. Thus, by repeating the concretization process using the learning model, the system concrete configuration can be derived efficiently.

[0073] In this example embodiment, a learning apparatus that generates a learning model for automatically designing a system as mentioned above, and a graph conversion device that solves the abovementioned problem by converting the learning data of the learning apparatus will be provided. In addition, the graph conversion device is a device that solves the problem described at the beginning of the present disclosure by converting the content of data to be learned by the learning apparatus.

[0074] Hereinafter, a graph conversion device in this example embodiment will be described. After that, an automated system design learning apparatus including the graph conversion device will be described. In the drawings described below, elements having the same functions, or the corresponding functions may be denoted by the same reference signs, and repeated description thereof may be omitted.

[Configuration]

[0075] First, a configuration of the graph conversion device in the present disclosure will be presented, followed by a configuration of the automated system design learning apparatus that is a learning apparatus generating a learning model automatically designing a system and that includes the above graph conversion device as part of a function of the learning apparatus.

[0076] Using FIG. 6, a configuration of an information processing system having the graph conversion device 10 in the first example embodiment will be described. FIG. 6 is a diagram showing an example of the information processing system.

[0077] As shown in FIG. 6, the information processing system includes the graph conversion device 10, a storage device 20, an input device 30, and an output device 40. The graph conversion device 10, the storage device 20, the input device 30, and the output device 40 are communicatively connected via a network.

[0078] The graph conversion device 10 is an information processing device such as a circuit, a server computer, a personal computer or a mobile device that is equipped with

a CPU (Central Processing Unit), a programmable device such as an FPGA (Field-Programmable Gate Array), a GPU (Graphics Processing Unit) or at least one or more of the above, for example.

[0079] The storage device **20** is a circuit or the like including a database, a server computer, and a memory. The storage device **20** stores a variety of information to be described later. In the example of FIG. **6**, one storage device **20** is provided outside the graph conversion device **10**, but a plurality of storage devices **20** may be provided inside or outside the graph conversion device **10**.

[0080] The input device **30** is a device such as a keyboard, a mouse, or a touchscreen, for example. The input device **30** is used for operating the graph conversion device **10**, the output device **40**, and so forth. The output device **40** is a display device such as a display, for example.

[0081] The communication network is a general communication network constructed using communication lines, such as the Internet, a LAN (Local Area Network), a dedicated line, a telephone line, an in-house network, a mobile communication network, Bluetooth (registered trademark), and Wi-Fi (Wireless Fidelity), for example.

[0082] The graph conversion device **10** is a device that converts the content of learning data used for learning by the automated system design learning apparatus **50** (described later) that designs an ICT system. As shown in FIG. **6**, the graph conversion device **10** includes a configuration collecting unit **11**, a feature value generating unit **12**, a distributed representation learning unit **13**, and a converting unit **14**. The respective functions of the configuration collecting unit **11**, the feature value generating unit **12**, the distributed representation learning unit **13** and the conversing means **14** can be realized by execution of a program for realizing the respective functions stored in the storage device by the arithmetic logic unit such as the CPU equipped in the information processing device configuring the graph conversion device **10**. The respective components will be described below. The overview of each component will be described first, and the details thereof will be described later in the operation description.

[0083] The configuration collecting unit **11** is a means for collecting system configurations. To be specific, the configuration collecting unit executes, on a system requirement that is information representing a configuration of a system including an abstract part (element), a concretization process of converting the abstract part into a concrete part, generates a system concrete configuration that is information representing a configuration of the system including no abstract part, generates design path information representing a process of generating the system concrete configuration from the system requirement (information including system requirement, system configuration proposal in the process of generating the system configuration proposal from the system requirement, and system concrete configuration), and stores all the configurations included in the design path information.

[0084] The feature value generating unit **12** is a means that extracts information necessary for learning by the graph conversion device **10** from the system configurations collected by the configuration collecting unit **11** and generates a feature value of a component. The feature value generating unit **12** includes a type identification information extracting unit **121**, a property information extracting unit **122**, and a graph structure extracting unit **123**.

[0085] The type identification information extracting unit **121** is a means for extracting type identification information that is information uniquely expressing components included in a system configuration, for each component.

[0086] The property information extracting unit **122** is a means for extracting information on a property (attribute) assigned to each of the components included in the system configuration.

[0087] The graph structure extracting units **123** is a means for extracting a structure when the system configuration is expressed by a graph (information on which elements are connected to each other).

[0088] The distributed representation learning unit **13** is a means that learns a distributed representation corresponding to each of the components included in the system configuration based on the information extracted by the feature value generating unit **12** (type identification information of element, property information of element, and information on configuration when system configuration is expressed by graph) and outputs a parameter necessary for calculating a distributed representation.

[0089] Here, "distributed representation", "parameter for calculating distributed representation", and "learning method for acquiring parameter" will be described.

[0090] First, a distributed representation corresponding to each component is a representation that expresses each component in a low-dimensional real-valued vector, and is information representing a feature of each component.

[0091] When comparing the distributed representations of certain two components, it can be determined that the closer the values of the respective constituents are, the similar the distributed representations of the two are. That is to say, in this case, it can be determined that the two components are similar components.

[0092] Conversely, it can be determined that the farther the values of the respective constituents of the distributed representations of the two components are, the less similar those two distributed representations are, and the two components are not similar components.

[0093] A distributed representation corresponding to a certain component is obtained by repeating an operation of adding the sequence of numerical values representing the feature value of the component and the sequence of numerical values representing the feature value of a component adjacent to the component with an appropriate weight, and updating the feature value of the component using the result of the addition.

[0094] A parameter necessary for calculating the distributed representation is here the weight described above. The parameter necessary for calculating the distributed representation determines the weight used for adding the feature value of a certain component and the feature value of a component adjacent to the component.

[0095] The parameter necessary for calculating the distributed representation mentioned above can be obtained by machine learning. A learning model to be machine-learned is a graph neural network that receives graph information in which each node and each edge in a configuration represented by a graph have feature values as an input, and that outputs graph information in the same format.

[0096] The abovementioned learning model receives a configuration in which a feature value is associated with each component as an input, then calculates a distributed representation of each component using the parameter, and

finally outputs a predicted value of a label corresponding to each component based on the distributed representation. In the learning, the parameter is optimized so that the error between the predicted value of the label and a truth label is minimized. In addition, the truth label may be, for example, a sequence of numerical values that can uniquely identify a component, but is not limited thereto.

[0097] The above is the description of "distributed representation", "parameter for calculating distributed representation" and "learning method for acquiring parameter" herein.

[0098] The converting unit **14** is a means that converts the system configuration input by the input device **30**, using the parameter output as a result of the learning by the distributed representation learning unit **13**, and outputs via the output device **40**. To be specific, the converting unit **14** calculates distributed representations of the respective elements included in the system configuration by using the parameter obtained from the distributed representation learning unit **13**, and assigns the distributed representations to the respective elements, thereby generating a configuration with distributed representation.

[0099] Next, using FIG. **7**, a configuration of the automated system design learning apparatus **50** that is a learning apparatus generating a learning model automatically designing a system, and that includes the graph conversion device **10** as part of a function of the learning apparatus in the first example embodiment will be described. FIG. **7** is a diagram showing an example of an information processing system.

[0100] As shown in FIG. **7**, the information processing system includes the automated system design learning apparatus **50**, a storage device **60**, an input device **70**, and an output device **80**. The automated system design learning apparatus **50**, the storage device **60**, the input device **70**, and the output device **80** are communicatively connected via a network.

[0101] The automated system design learning apparatus **50** is, for example, an information processing apparatus such as a circuit, a server computer, a personal computer, or a mobile terminal, which is equipped with a CPU (Central Processing Unit), a programmable device such as an FPGA (Field-Programmable Gate Array), a GPU (Graphic Processing Unit), or one or more of the above.

[0102] The storage device **60** is a database, a server computer, a circuit having memory, and so forth. The storage device **60** stores various information, which will be described later. In the example of FIG. **7**, one storage device **60** is provided outside the automated system design learning apparatus **50**, but a plurality of storage devices **60** may be provided inside or outside the automated system design learning apparatus **50**.

[0103] The input device **70** is, for example, a device such as a keyboard, a mouse, and a touchscreen. The input device **70** is used when operating the automated system design learning apparatus **50**, the output device **80**, and so forth. The output device **80** is, for example, a display device such as a display.

[0104] The communication network is, for example, a general communication network constructed using communication lines such as the Internet, a LAN (Local Area Network), a dedicated line, a telephone line, an in-house network, a mobile communication network, Bluetooth (registered trademark), and Wi-Fi (Wireless Fidelity).

[0105] The automated system design learning apparatus **50** is an apparatus that learns a learning model designing an ICT system. As shown in FIG. **7**, the automated system design learning apparatus **50** includes a designing unit **51**, the graph conversion device **10**, a reward setting unit **52**, a learning data generating unit **53**, and a learning unit **54**. The respective functions of the design unit **51**, the graph conversion device **10**, the reward setting unit **52**, the learning data generating unit **53**, and the learning unit **54** can be realized by execution of a program for realizing the respective functions stored in the storage device by the arithmetic logic unit such as the CPU equipped in the information processing apparatus configuring the automated system design learning apparatus **50**. The respective components will be described below. The overview of each component will be described first, and the details thereof will be described later in the operation description.

[0106] The design unit **51** executes, on a system requirement that is information representing a configuration of a system including an abstract part (element), a concretization process of converting the abstract part into a concrete part, generates a system concrete configuration that is information representing a configuration of a system including no abstract part, and generates design path information representing a process of generating the system concrete configuration from the system requirement (information including system requirement, system configuration proposal in the process of deriving system concrete configuration from the system requirement, and system concrete configuration).

[0107] The graph conversion device **10** is a device that converts the content of learning data used for learning by the automated system design learning apparatus **50** that designs an ICT system as described above. How the graph conversion device **10** operates in the automated system design learning apparatus **50** will be described in detail in the operation description later.

[0108] The reward setting unit **52** sets a reward for each of the system requirement with distributed representation, the system configuration proposal with distributed representation and the system concrete configuration with distributed representation that are included in the design path information with distributed representation. Moreover, the reward setting unit **52** sets a reward for learning a system designing method based on the requirement.

[0109] The learning data generating unit **53** generates learning data in which the reward is associated with each of the system requirement with distributed representation, the system configuration proposal with distributed representation and the system concrete configuration with distributed representation that are included in the design path information with distributed representation.

[0110] The learning unit **54** learns a system designing method based on the requirement, based on the learning data.

[Operation]

[0111] Next, the operation of the graph conversion device **10** and the automated system design learning apparatus **50** including the graph conversion device **10** in the first example embodiment will be described. In the following description, the drawings will be referred to as necessary. In the first example embodiment, an automated system design learning method is executed by causing the graph conversion device **10** and the automated system design learning

apparatus **50** to operate. Therefore, the description of the automated system design learning method in the first example embodiment is equal to the description of the operation of the automated system design learning apparatus including the graph conversion device below.

[0112] First, the operation of the graph conversion device **10** will be described, followed by the operation of the automated system design learning apparatus **50** including the graph conversion device **10**.

[0113] FIG. **8** is a diagram for describing an example of the operation of the graph conversion device **10** in the first example embodiment. First, the configuration collecting unit **11** collects system configurations as information to be the source of learning data for learning a distributed representation corresponding to each element of a configuration (step S**1**).

[0114] FIG. **9** is a diagram for describing an example of the operation of collecting system configurations for distributed representation learning (operation at step S**1** described above).

[0115] First, the configuration collecting unit **11** performs system design, and generates design path information (step S**11**). For generating design path information, for example, a design function (designing unit **51**) of the automated system design learning apparatus **50** can be used.

[0116] To be specific, the configuration collecting unit **11** executes, on a system requirement that is information representing a system configuration including an abstract part (element), a concretization process of converting the abstract part into a concrete part, generates a system concrete configuration that is information representing a system configuration including no abstract part, and generates design path information representing a process of generating the system concrete configuration from the system requirement (information including the system requirement, system configuration proposal in the process of deriving the system concrete configuration from the system requirement, and system concrete configuration).

[0117] Herein, it will be described as using the design function of the automated system design learning apparatus **50** for generating the design path information.

[0118] FIG. **10** is a diagram for describing the operation of generating the design path information.

[0119] First, the configuration collecting unit **11** acquires a system requirement stored in advance in the storage device **20**, and sets the acquired system requirement as a current configuration (step D**1**). Next, the configuration collecting unit **11** registers the current configuration as a root node of a search tree (step D**2**). To be specific, the configuration collecting unit **11** sets system requirement identification information for identifying the current configuration (system requirement) as the root node, and stores it into search tree information. The data structure of the search tree information will be described later with reference to FIG. **12**.

[0120] Here, FIG. **11** is a diagram for describing an example of the search tree. The search tree can be represented by a graph (search tree T**1**) as shown in FIG. **11**. The root node referred to in the description of step D**2** described above corresponds to the system requirement R**1** in FIG. **11**.

[0121] FIG. **12** is a diagram for describing an example of data structure of the search tree information. The data structure shown in FIG. **12** corresponds to the search tree of FIG. **11**. In the example of FIG. **12**, system requirement identification information "R**1**" associated with the system requirement R**1** stored in the storage device **20** is stored in "parent node" of the search tree information T**10**, and moreover, "1" is stored as information indicating that the system requirement R**1** is "root node".

[0122] Next, in a case where no concretization rule can be applied to the current configuration or in a case where the current configuration is a system concrete configuration (step D**3**: Yes), the configuration collecting unit **11** proceeds to a process at step D**9**.

[0123] Further, in a case where there is a concretization rule applicable to the current configuration (No: step D**3**), the configuration collecting unit **11** repeats the processing from step D**4** to step SD**8**.

[0124] Next, the configuration collecting unit **24** concretizes one of components included in the current configuration by executing a concretization process (step D**4**). Next, the configuration collecting unit **11** sets a configuration obtained by concretizing the abovementioned component at step D**3** as a next configuration (step D**5**).

[0125] Next, in a case where the next configuration (concretized configuration (system configuration proposal)) is not stored in the search tree information (step D**6**: No), the configuration collecting unit **11** proceeds to a process at step D**7**. In a case where the next configuration (concretized configuration (system configuration proposal)) is stored in the search tree information (step D**6**: Yes), the configuration collecting unit **11** proceeds to a process at step D**8**.

[0126] Next, with a node representing the current configuration as a parent node, the next configuration (concretized configuration) as a child node, and an edge representing the concretized component (concretized component of concretization rule) as a directed edge, the configuration collecting unit **11** associates the next configuration and the concretized component with the current configuration, and stores into the search tree information (step D**7**).

[0127] For example, in a case where the current configuration is the system requirement R**1** and the system configuration proposal R**2** is generated from the system requirement R**1**, system configuration proposal identification information "R**1**" corresponding to the system requirement R**1** in the search tree information T**10**, system configuration proposal identification information "R**2**" corresponding to the system configuration proposal R**2** generated through the concretization process, and concretization identification information "Rule**1**" corresponding to the concretization rule used in the concretization process are associated and stored as shown in FIG. **12**.

[0128] Further, in a case where the current configuration is the system configuration proposal R**2** and a concrete configuration R**5** is generated from the system requirement proposal R**2**, the system configuration proposal identification information "R**2**" of the search tree information T**10**, the system concrete configuration identification information "R**5**" associated with the concrete configuration generated through the concretization process, and the concretization rule identification information "Rule**4**" associated with the concretization rule used in the concretization process are associated and stored as shown in FIG. **12**.

[0129] Next, the configuration collecting unit **11** sets the next configuration as a current configuration (step D**8**).

[0130] Next, the configuration collecting unit **11** generates a series of paths (design path information that stores configurations in chronological order) up to generation of the current system concrete configuration the system require-

ment, based on the system configurations (system requirement, system configuration proposal, system concrete configuration) obtained in the processing from step D1 to step D8 described above (step D9). Moreover, there is a case where no applicable concretization rule is left before the processing from step D1 to step D8 is repeated and the system concrete configuration is reached, and the design becomes a deadlock. In this case, the configuration collecting unit **11** generates a series of paths from the system requirement to the last reached system configuration proposal.

[0131] FIG. **13** is a diagram for explaining an example of the data structure of the design path information. In design path information DP1 shown in FIG. **13**, the system requirement "R1", the system configuration proposal "R2", and the system concrete configuration "R5" shown in FIG. **11** are recorded in this order. That is to say, the design path information DP1 shows that, as one of the processes in system design, there is a series of design processes starting from the system requirement "R1", passing through the system configuration proposal "R2", and leading to the system concrete configuration "R5".

[0132] Each of the configurations associated with the design path information (configurations such as R1, R2, and R5) is information replaced by a configuration with distributed representation generated by rewriting information held by each of the elements, as will be described later, and then used for generating learning data.

[0133] In the above manner, after generating the design path information (step S11 of FIG. **9**), the configuration collecting unit **11** stores all the configurations included in the generated design path into the storage device (step S12).

[0134] Next, in a case where the design path information is generated for a designated number of times (step S13: Yes), the configuration collecting unit **11** passes all the configurations stored in the storage device to the feature value generating unit (step S14). In a case where the configuration collecting unit **11** has not generated the design path information for a designated number of times (step S13: No), the configuration collecting unit **11** performs the processing from step S11 to step S13 again.

[0135] The number of times the design path information is generated set in advance is determined by, for example, experiments, simulations, and the like.

[0136] In this manner, the configuration collecting unit **11** collects system configurations as information to be the source of learning data for learning a distributed representation corresponding to each of the elements of the configuration, and passes it to the feature value generating unit **12** (step S1).

[0137] Next, the feature value generating unit **12** extracts information necessary for learning a distributed representation of each component included in the configuration, from the received system configuration (step S2 of FIG. **8**). FIG. **14** shows an example of operation in which the feature value extracting unit **12** extracts information necessary for learning a distributed representation of each component and generates a feature value of the component.

[0138] First, the feature value generating unit **12** selects any configuration from among the configurations received from the configuration collecting unit **11** and sets it as a current configuration (step S21). Next, the feature value generating unit **12** selects one of components having not been selected from among components included in the current configuration (step S22).

[0139] Next, the type identification information extracting unit **121** acquires type identification information of the selected component (step S23). The type identification information is information that can uniquely identify the type of a component. For example, a one-hot vector can be considered as the type identification information. The one-hot vector is a vector in which one element is "1" and the other elements are "0", and the respective elements of the vector correspond to the respective types one to one.

[0140] FIG. **15** is a diagram showing an example of the data structure of the type definition information and an example of types and identification information corresponding to the types. Reference numeral **151** in FIG. **15** denotes the type definition information. This shows type definition information used in design of the face recognition system shown in FIG. **1**. In the example of the type definition information **151** shown in FIG. **15**, the type names of "camera", "face recognition software", "server computer", "server computer A", "server computer B", "connect", "HTTP communication" and "join" representing the types are stored as "type (function information)" representing a type.

[0141] In the example of FIG. **15**, the types used in the description of the face recognition system in FIG. **1** are shown as an example of the types stored, but the types stored in the type definition information are not limited to the types described in FIG. **15**, and the types may be defined as types other than the types described in FIG. **15**.

[0142] Further, reference numeral **152** in FIG. **15** is information representing the type identification information. The type identification information **152** is an example of a data structure of the type identification information corresponding to the type definition information **151** of FIG. **15** likewise. The type identification information **152** is information in which the type information representing the type is associated with an array having "0" and "1" in the elements as identification information corresponding to the type information. Such a data structure is generally referred to as a one-hot vector. The length of each array included by the type identification information shall be at least the same as or greater than the number of types stored in the type definition information. In addition, the elements of each array included by the type identification information correspond to the types stored in the type definition information in the order in which they are defined.

[0143] For example, in the case of type "camera" defined first in the type definition information, in an array (array "camera") corresponding to "camera" in the type identification information, "1" is stored in the first element and "0" is stored in the remaining elements. For each of the types other than "camera", an array in which "1" is stored in a position corresponding to the type and "0" is stored in the remaining positions is associated and stored.

[0144] Next, the property information extracting unit **122** acquires property information of the selected component (step S24). The property information is information representing a characteristic or attribute associated with the type of each component. FIG. **16** shows an example of the type property information showing a type and a property defined for the type, and a property list that lists all the properties defined.

[0145] Reference numeral **161** of FIG. **16** is an example of a type and the data structure of type property information representing a property defined for the type.

[0146] For example, in the case of type "camera" defined first in the type property information **161**, a property corresponding to "camera" of the type property information **161** is "cost (cost for purchase)" and "quality (image quality)".

[0147] Likewise, for a type other than "camera", it is possible to refer to a property defined for the type.

[0148] Further, reference numeral **162** in FIG. **16** is a diagram for showing an example of the data structure of the property list. The property list **162** is information in which all the properties defined in the type property information are recorded without excess or deficiency. The properties recorded in the property list are not duplicated, and the order in which the properties are recorded does not change unless there is a change in the type and property defined in the type definition information.

[0149] Next, the property information extracting unit **122** generates a property vector from the type property information and the property list (step S**25**). FIG. **17** is a diagram for showing an example of a property vector generated from the type property information and the property list.

[0150] Reference numeral **171** of FIG. **17** is a diagram that lists all property vectors that can be generated from the type property information **161** and the property list **162** shown in FIG. **16** for simplicity. Each of the elements of the property vector corresponds to each of the elements of the property list as shown in FIG. **16**. That is to say, the length of the property vector is the same as the length of the property list. To be specific, the property vector is information in which the type information representing the type is associated with an array having elements in which the presence or absence of a property corresponding to the type information is represented by "0" or "1". Such a data structure is generally referred to as a one-hot vector. At this time, the elements of each array included by the property vector correspond in the order of the properties stored in the property list.

[0151] For example, a property vector corresponding to "camera" shown first in the property vector information **171** is a vector in which the first and second elements are "1" and all the remaining elements are "0". Referring to the properties defined in "camera" of the type property information **161**, they are "cost" and "quality". Here, referring to the property list **162**, "cost" is recorded first, and "quality" is recorded second. That is to say, the fact that the first and second elements of the property vector of "camera"" shown in FIG. **17** are "1" indicates that the properties "cost" and "quality" are defined for type "camera".

[0152] Although the property vector corresponding to type "camera" has been described in the above description, the property vector can be generated for any type other than type "camera".

[0153] Next, the feature value generating unit **12** combines the type identification information and the property vector, and assigns this to the component as a feature value of the component (step S**26**). FIG. **18** is a diagram for showing information generated by combining the type identification information and the property vector. For simplicity, all the information that can be generated by the abovementioned method will be shown in a list. Information **181** in FIG. **18** is a combination of included information for each type based on the type identification information **152** shown in FIG. **15** and the property vector information **171** shown in FIG. **17**.

[0154] Of the information **181** shown in FIG. **18**, a part corresponding to the type identification information is shown surrounded by a dotted line. Taking type "camera" as an example, the type identification information of type "camera" is a vector in which the first element is "1" and the second to eighth elements are "0", so that the element of a vector corresponding to type "camera" in the information **181** is like that. In addition, the property vector of type "camera" is a vector in which the first element and the second element are "1" and the subsequent elements are "0". In the example shown in FIG. **18**, the property vector is combined to the end of the type identification information, so that the ninth and tenth of the generated vector are "1" and the subsequent elements are "0".

[0155] The feature value generating unit **12** does not generate the information **181** shown in FIG. **18** as it is. The information **181** in FIG. **18** is merely an example, showing a list of all information that can be generated for each type defined in the type definition information **151** in FIG. **15**.

[0156] The feature value generating unit **12** assigns information generated by combining the type identification information and the property vector for each component in the manner as described above to the component as a feature value of the component.

[0157] Next, in a case where all the components of the current configuration have not been selected (step S**27**: No), the feature value generating unit **12** proceeds to the process at step S**22**, and executes the processes from step S**22** to step S**27** again. In a case where all the components of the current configuration have been selected (step S**27**: Yes), the feature value generating unit **12** proceeds to a process at step S**28**.

[0158] Next, the graph structure extracting unit **123** acquires the graph structure information of the current configuration (step S**28**). The graph structure information is information representing which components (nodes) are connected to each other by a relation (edge) in a configuration. The data format of the graph structure information is the same as the format shown in FIG. **3**, and is information including the component identification information of all the nodes present in the configuration, the component identification information of all the edges, and the component identification information of the starting node and the component identification information of the end node of all the edges.

[0159] Next, in a case where all the configurations received from the configuration collecting unit **11** have not been selected (step S**29**: No), the feature value generating unit **12** proceeds to the process at step S**210**. In a case where all the configurations have been selected (step S**29**: Yes), the feature value generating unit **12** proceeds to the process at step S**211**.

[0160] The feature value generating unit **12** selects a configuration that has not been selected yet and sets it as a current configuration in the process at step S**210**. Upon completion of the process at step S**210**, the feature value generating unit **12** proceeds to the process at step S**22**, and executes the subsequent process again.

[0161] Finally, the feature value generating unit **12** passes all the configurations to a distributed representation learning unit at step S**211**. At the time of the process at step S**210**, in all the configurations passed to the distributed representation

learning unit, to all the components included in those configurations, "information combining the type identification information and the property vector" corresponding thereto are assigned. Moreover, by the process at step S**28**, graph structure information of each configuration is associated with the configuration.

[0162] By performing the above-described operation, the graph conversion device **10** uses the feature value generating unit **12** to extract information necessary for learning a distributed representation and generates the feature value of a component.

[0163] Next, the distributed representation learning unit **13** performs learning using all the configurations received from the feature value generating unit **12** and the graph structure information associated with the respective configurations, and acquires a parameter necessary for calculating a distributed representation (step S**3**). Here, a learning model to be learned is a Graph Neural Network (GNN) that receives, as an input, graph information in which each node and each edge in a configuration represented by a graph have feature values, and that outputs graph information in the same format. As a learning method, a method aimed at acquiring a distributed representation of elements configuring a graph by learning, such as the techniques described in Non-Patent Literature 1 and Non-Patent Literature 2, is preferable, but is not limited to a specific learning method.

[0164] The description of the basic learning method that is not limited to a specific method is as described above for the distributed representation learning unit **13** with reference to FIG. **6**.

[0165] To be specific, for example, a method can be considered where: each node (component) is given a feature value composed of type identification information and a property vector; after a distributed representation of each node is calculated in consideration of the relation and adjacency information between nodes, a model to predict the component identification information of an adjacent node based on that is prepared; and then learning is performed to minimize the binary cross-entropy error between the predicted value and numerical values representing the actual component identification information (for example, a vector with elements of 0 and 1).

[0166] Finally, the converting unit **14** stores the parameter acquired by the learning by the distributed representation learning unit **13**, necessary for calculating a distributed representation.

[0167] The converting unit **14** having received the parameter necessary for learning a distributed representation can calculate a distributed representation of each component included in graph data (one or more system configurations) input through the input device **30**. Moreover, the converting unit **14** can generate a configuration with distributed representation by assigning the calculated distributed representation of each component to the component, and output it through the output device **40**.

[0168] As described above, the graph conversion device **10** can learn a parameter for calculating a distributed representation of each component by extracting necessary information from a configuration and using it as learning data, generate a configuration with distributed representation by assigning the distributed representation of each component of the input graph data to the component, and output the configuration with distributed representation.

[0169] Next, the operation of the automated system design learning apparatus **50** including the graph conversion device **10** will be described.

[0170] Herein, the graph conversion device **10** is a device incorporated inside the automated system design learning apparatus **50** for the purpose of reducing time costs since the automated system design learning apparatus **50** makes a change to learning data used for learning so that relearning of the system design apparatus that would have been necessary if a new element had been added is not necessary. The change to be made to the learning data by the automated system design learning apparatus **50** is to assign a distributed representation corresponding to a component included by a configuration to the component. The graph conversion device **10** shall have completed learning so as to be able to calculate an appropriate distributed representation before the automated system design learning apparatus **50** is operated.

[0171] Further, the automated system design learning apparatus **50** is a device that makes system design more efficient by learning so as to be able to select an appropriate concretization rule from among the applicable concretization rules described above. Based on the above, the operation of the automated system design learning apparatus **50** including the graph conversion device **10** will be described.

[0172] FIG. **19** is a diagram for showing an example of the operation of the automated system design learning apparatus **50** including the graph conversion device **10** in the first example embodiment. First, the automated system design learning apparatus **50** acquires a system requirement to be a learning target from the storage device **60** (step S**31**).

[0173] Next, the automated system design learning apparatus **50** executes reinforcement learning during a preset period using the learning target system requirement (step S**32**). Consequently, a learning model is generated. For example, in a case where a neural network is used for reinforcement learning, the learning model is the neural network.

[0174] The preset learning period is determined by, for example, experiments, simulations, and the like. In addition, the learning period may be the number of times of learning, instead of the learning period. The number of times of learning is, for example, the number of updates of the weight of the neural network when the neural network is used.

[0175] Here, the details of the learning operation that is the above process at step S**32** will be described later with reference to FIG. **20**. In the learning, an ICT system designing method will be learned.

[0176] First, the designing unit **51** generates design path information using the system requirement (step S**321**). To be specific, the designing unit **51** generates design path information as shown in the flowchart of FIG. **10**. Since the operation of the designing unit **51** to generate the design path information is the same as that described using FIG. **10** as the description of the example of the operation of the configuration collecting unit **11** mentioned above, the description will be omitted here.

[0177] After the designing unit **51** generates the design path information (step S**321** of FIG. **20**), the automated system design learning apparatus **50** uses the graph conversion device **10** to convert it into design path information with distributed representation (step S**322**). To be specific, the graph conversion device **10** generates design path information with distributed representation as shown in a flowchart of FIG. **21**. In addition, the process of generating the design

path information with distributed representation by the graph conversion device **10** to be described below is performed for each path, which is a series of design processes.

[0178] First, the graph conversion device **10** set the first configuration of the design path information as a current configuration (step S**3221**). Next, the graph conversion device **10** selects a component having not been selected from among components included by the configuration (step S**3222**).

[0179] Next, the graph conversion device **10** calculates a distributed representation of the selected component and assigns it to the component (step S**3223**). Assigning a distributed representation to a component means enabling the distributed representation of the component to be referenced. Next, in a case where all the components of the current configuration have not been selected (step S**3224**: No), the graph conversion device **10** proceeds to the process at step S**3222**, and executes the processes from step S**3222** to step S**3224** again. In a case where all the components of the current configuration have been selected (step S**3224**: Yes), the graph conversion device **10** proceeds to a process at step S**3225**.

[0180] Next, in a case where all the configurations of the design path information have not been selected (step S**3225**: No), the graph conversion device **10** proceeds to a process at step S**3226**. At step S**3226**, the graph conversion device **10** sets the next configuration of the design path information as a current configuration. After that, the graph conversion device **10** proceeds to the process at step S**3222**. In a case where all the configurations of the design path information have been selected (step S**3225**: Yes), the graph conversion device **10** proceeds to a process at step S**3227**.

[0181] Finally, the graph conversion device **10** generates design path information with distributed representation (step S**3227**).

[0182] FIG. **22** is a diagram for describing an example of the data structure of the design path information with distributed representation. Design path information with distributed representation EDP**1** shown in FIG. **22** represents a configuration in which a distributed representation is given to a component of each of the configurations associated with the design path information DP**1** shown in FIG. **13**. That is to say, a system requirement with distributed representation "ER**1**", a system configuration proposal with distributed representation "ER**2**", and a system concrete configuration with distributed representation "ER**5**" shown in FIG. **22** refer to configurations in which distributed representations are assigned to the components of the system requirement "R**1**", the system configuration proposal "R**2**", and the system concrete configuration "R**5**".

[0183] Thus, the graph conversion device **10** converts design path information to design path information with distributed representation (step S**322** of FIG. **20**).

[0184] Next, the reward setting unit **52** sets a reward for each system configuration with distributed representation of the design path information with distributed representation (the system requirement with distributed representation, or the system configuration proposal with distributed representation, or the system concrete configuration with distributed representation, which are mentioned above) (step S**323** of FIG. **20**). To be specific, the reward setting unit **52** sets a reward as shown in a flowchart of FIG. **23**. A reward setting process by the reward setting unit **52** to be described below is performed for each path, which is a series of design processes.

[0185] Here, in the system design described in the present disclosure, it is important to apply an appropriate concretization rule to a configuration proposal and efficiently reach a system concrete configuration. Therefore, a process of giving a probability that each configuration proposal can reach a concrete configuration as a reward by using design path information with distributed representation is performed.

[0186] First, the reward setting unit **52** determines the initial value of an update candidate reward (step S**41**). Specifically, in a case where the last configuration of the design path information with distributed representation is a system concrete configuration with distributed representation, the reward setting unit **52** sets, as the initial value of the update candidate reward, the reward of the system concrete configuration with distributed representation to "1", for example, and in a case where it is not a system concrete configuration with distributed representation, the reward setting unit **52** sets the reward corresponding to the last configuration to "0", for example.

[0187] Here, the update candidate reward described above is a value used when updating the value of a reward obtained by each system configuration (system configuration with distributed representation) of the design path information with distributed representation at step S**42**, which will be described later.

[0188] In the case of the example of FIG. **22**, the last configuration of the design path information with distributed representation EDP**1** is the system concrete configuration with distributed representation "ER**5**", so that the reward "1" is associated with "ER**5**" representing the system concrete configuration with distributed representation of the design path information with distributed representation EDP**1** in the process at step S**41**.

[0189] Next, the reward setting unit **52** updates the reward of each system configuration (system configuration with distributed representation) included in the design path information with distributed representation (step S**42**). Specifically, the reward setting unit **52** performs a process to update the reward as shown in a flowchart of FIG. **24**.

[0190] First, the reward setting unit **52** sets the last configuration (system configuration with distributed representation) in the design path information with distributed representation as an update target configuration (step S**51**). Next, the reward setting unit **52** compares a reward associated with the update target configuration with an update candidate reward, selects the larger reward, and stores the selected reward in association with the update target configuration (step S**52**). Next, the reward setting unit **52** sets the reward selected at step S**52** (the larger reward in comparison) as the update candidate reward (step S**53**).

[0191] Next, in a case where the reward update process has not been performed on all the configurations in the design path information with distributed representation (in a case where execution of the reward update process has not finished to the first configuration in the design path information with distributed representation) (step S**54**: No), the reward setting unit **52** sets a configuration immediately preceding the current update target configuration as an update target configuration (step S**55**), and proceeds to the process at step S**52**.

[0192] Further, in a case where the reward update process has been executed on all the configurations in the design path information with distributed representation (in a case where the reward update process has finished to the first configuration in the design path information with distributed representation) (step S**54**: Yes), the reward setting unit **52** ends the process at step S**42** shown in FIG. **24**.

[0193] To be specific, as the update candidate reward to be set at step S**42** is, after comparison between the reward associated with the update target configuration and the evaluation of the update candidate, it is regarded as "0" if the former reward is not present yet (if it is not associated) before the comparison. Then, after the comparison, the update is performed by setting the larger one as the reward associated with the update target configuration and the update candidate reward. After that, the update target configuration is updated, and this process is repeated.

[0194] For example, a reward is not associated with any of the configurations at first, so that in the example of FIG. **22**, "ER**5**" is the first update target configuration at step S**51**. Since a reward has not been associated with "ER**5**" yet, its value is considered to be "0". In that case, the update candidate reward has been initialized to "1" at step S**41**, so that "0" and "1" are compared with each other, and the larger reward "1" is associated with "ER**5**".

[0195] Next, "ER**2**" is set as an update target configuration, and the same processing as the abovementioned processing is executed again. After that, further, "ER**1**" is set as an update target configuration, and the same processing as the abovementioned processing is repeated.

[0196] As another example, in a case where, starting as the first update candidate configuration (a configuration that is not a system concrete configuration with distributed representation), it is not a concrete configuration, the update candidate reward is initialized to "0", and in a case where a reward "1" is associated with the next configuration, "0" and "1" are compared and the value of the update candidate reward is updated to "1", which is larger.

[0197] Specifically, in the process at step S**42**, data (reward) for learning the evaluation values of the configurations "ER**2**" and "ER**1**" in the design process based on the evaluation value of the configuration "ER**5**" at the end of the design is generated from the process performed at step S**51**. At this time, if larger data (reward) has been obtained previously, it is given priority, so that it is compared with the reward stored in association with the configuration, and the larger one is left.

[0198] In the above description, the larger one is left and propagated upward ("ER**1**" when seen from "ER**2**") because it is known that a transition can be made from "ER**1**" to "ER**2**". What to be learned is an expected reward to be obtained when the best option is kept to be selected, and it is known that a transition can be made from "ER**1**" to "ER**2**" when an appropriate concretization rule is applied. Therefore, when the reward associated with "ER**2**" is greater, the reward of "ER**5**" should be propagated to "ER**1**" as a priority.

[0199] A transition from "ER**1**" to "ER**2**" does not necessarily occur, but there is a possibility thereof. For example, in the example of FIG. **11**, there is a possibility of transition to "ER**3**" or "ER**4**". However, the automatic design function can select which transition to make, so that a larger reward in comparison is propagated upward.

[0200] After setting the reward in the above manner (step S**323** in FIG. **20**), the learning data generating unit **53** generates learning data by associating the design system configuration with distributed representation and the reward for the design path information with distributed representation, and stores the generated learning data into the storage device **60** (step S**324** in FIG. **20**). Specifically, the learning data generating unit **53** generates learning data as shown in a flowchart of FIG. **25**.

[0201] First, the learning data generating unit **53** sets the first configuration of the design path information with distributed representation as a configuration with distributed representation to be a learning data generation target (step M**11**). Next, the learning data generating unit **53** sets the first configuration of the design path information with distributed representation as a configuration to be a reward reference target (step M**12**).

[0202] Next, the learning data generating unit **53** generates learning data by pairing the configuration with distributed representation to be the target for generating learning data and the reward corresponding to the configuration (the reward set at step **52** of FIG. **24**), and stores the generated learning data into the storage device **60** (step M**13**).

[0203] Next, in the case of having not generated learning data for all the configurations included in the design path information with distributed representation (in the case of having not finished generating learning data to the last configuration in the design path information with distributed representation) (step M**14**: No), the learning data generating unit **53** proceeds to a process at step M**15**. In the case of having generated learning data for all the configurations included in the design path information with distributed representation (in the case of having finished generating learning data to the last configuration in the design path information with distributed representation) (step M**14**: Yes), the learning data generating unit **53** ends the process to generate learning data.

[0204] Next, the learning data generating unit **53** sets the next configuration in the design path information with distributed representation as a learning data generation target configuration (step M**15**).

[0205] Next, the learning data generating unit **53** sets the next configuration in the design path information with distributed representation as a reward reference target configuration (step M**16**).

[0206] After the learning data is generated in the above manner (step S**324** of FIG. **20**), the learning unit **54** performs learning based on the learning data (step S**325** of FIG. **29**). Specifically, the learning unit **54** performs learning of a learning model as shown in a flowchart of FIG. **26**.

[0207] Here, a learning model to learn is a GNN (Graph Neural Network) that receives graph information in which each node and edge in a graph and the graph itself have attribute values as an input and outputs graph information in the same format.

[0208] First, the learning unit **54** selects one learning data D**1** from among the learning data generated as described above (step L**11**). Next, the learning unit **54** inputs graph information representing a configuration with distributed representation included by the learning data D**1** into a first learning model, and acquires a result (graph information OG**1**) output from the learning model (step L**12**).

[0209] Next, the learning unit **54** generates a pair VP**1** by pairing an attribute value At**1** of the graph itself and a reward included by the data D**1** of the graph information OG**1** (step L**13**).

[0210] Next, in the case of having selected all the learning data (step L**14**: Yes), the learning unit **54** proceeds to a process at step L**15**. In the case of having not selected all the learning data (step L**14**: No), the learning unit **54** proceeds to the process at step L**11**, and executes the processes up to step L**13** again.

[0211] Next, the learning unit **54** generates pairs VP**1** for all the data included in the learning data and gathers as VPS**1** (step L**15**).

[0212] Then, the learning unit **54** learns the first learning model using the respective pairs included by VPS**1** (step L**16**). To be specific, the learning unit **54** uses the respective pairs included by VPS**1** to train the learning model so as to minimize a loss function with the mean squared error between the attribute value and the determined reward as the loss function.

[0213] As described above, during a learning period set in advance, the processes from step S**321** to step S**325** shown in FIG. **20** are repeated to perform reinforcement learning (step S**32** in FIG. **19**).

[0214] Then, using the learning model at the current moment generated through the abovementioned reinforcement learning, the automated system design learning apparatus **50** executes a concretization process (design) on the system requirement (step S**33** of FIG. **19**).

[0215] The concretization process (design) is a process to concretize an abstract part (component) included by a system requirement. By repeatedly executing the concretization process, it is possible to finally derive a system concrete configuration including only a concrete part. An example of the concretization process is as shown in FIG. **1**.

[0216] Subsequently, in a case where it is determined that learning is enough based on the result at step S**33** (step S**34**: Yes), the automated system design learning apparatus **50** ends reinforcement learning at step S**32**. In a case where it is determined that learning is not enough (step S**34**: No), the automated system design learning apparatus **50** repeats the processes at steps S**32** and S**33** until it is determined that learning is enough.

[0217] In the determination at step S**34**, it is determined to be enough, for example, when the number of steps of search is equal to or less than a preset threshold value A. Also, in the determination at step S**34**, it may be determined to be enough when the number of steps of search performed at step S**33** previously is equal to or less than a preset threshold value B consecutively for a specified number of times. However, the determination at step S**34** is not limited to the abovementioned methods.

[0218] The number of steps of search described above depends on the method for design at step S**33**. The number of steps of search is the number of times that a concretization rule is applied to a system requirement or a system configuration proposal until a system concrete configuration is derived from the system requirement. The number of steps of search can also be expressed as the number of system configuration proposals actually reached during the design.

[0219] As described above, according to this example embodiment, when a system design is performed by adding a new element, learning is performed by converting learning data into a format that allows evaluation of an unlearned element using experience on previously learned elements, so that relearning of the system design apparatus that would have been necessary if a new element had been added is not necessary, thereby reducing time costs.

Second Example Embodiment

[0220] Next, a second example embodiment of the present disclosure will be described with reference to the drawings. In this example embodiment, the overview of the configuration of the automated system design learning apparatus described in the above example embodiment will be shown. FIGS. **27** and **28** are diagrams for describing the configuration, and these drawings may be relevant to any of the example embodiments.

[0221] First, the hardware configuration of an information processing apparatus **100** will be described with reference to FIG. **27**. The information processing apparatus **100** is configured with a general information processing apparatus and, as an example, has the following hardware configuration including;

- **[0222]** a CPU (Central Processing Unit) **101** (arithmetic logic unit),
- **[0223]** a ROM (Read Only Memory) **102** (memory unit),
- **[0224]** a RAM (Random Access Memory) **103** (memory unit),
- **[0225]** programs **104** loaded to the RAM **103**,
- **[0226]** a storage device **105** storing the programs **104**,
- **[0227]** a drive device **106** performing reading from and writing into a storage medium **110** outside the information processing apparatus,
- **[0228]** a communication interface **107** connected to a communication network **111** outside the information processing apparatus,
- **[0229]** an input/output interface **108** inputting and outputting data, and
- **[0230]** a bus **109** connecting the respective components.

[0231] FIG. **27** shows an example of the hardware configuration of the information processing apparatus serving as the information processing apparatus **100**, and the hardware configuration of the information processing apparatus is not limited to the abovementioned case. The information processing apparatus may be configured with part of the above configuration, for example, may omit the drive device **106**. Moreover, the information processing apparatus can use, instead of the abovementioned CPU, a GPU (Graphic Processing Unit), a DSP (Digital Signal Processor), an MPU (Micro Processing Unit), an FPU (Floating point number Processing Unit), a PPU (Physics Processing Unit), a TPU (Tensor Processing Unit), a quantum processor, a microcontroller, a combination thereof, or the like.

[0232] Then, by causing the CPU **101** to acquire and execute the programs **104**, the information processing apparatus **100** can construct and include a configuration collecting unit **131**, a feature value generating unit **132**, a distributed representation learning unit **133**, and a converting unit **134** shown in FIG. **28**. The programs **104** are, for example, stored in advance in the storage device **105** or the ROM **102**, and are loaded to the RAM **103** and executed by the CPU **101** as required. The programs **104** may be provided to the CPU **101** via the communication network **111**, or may be stored in advance in the storage medium **110** and be read out by the drive device **106** and provided to the CPU **101**. However, the configuration collecting unit **131**, the feature

value generating unit **132**, the distributed representation learning unit **133**, and the converting unit **134** mentioned above may be constructed using dedicated electronic circuits for realizing such means.

[0233] The configuration collecting unit **131** collects system configuration information included by design path information representing a design process of the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element to a concrete element using a preset concretization rule. The feature value generating unit **132** generates a feature value of the element in the system configuration information included by the design path information. The distributed representation learning unit acquires, by machine learning, a parameter for calculating a distributed representation of the element by using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element. The converting unit generates design path information with distributed representation by calculating using the parameter the distributed representation of the element within the system configuration information included by the design path information, which is a learning target of a learning model machine-learned so as to convert an abstract element into a concrete element using the aforementioned concretization rule, and assigning the distributed representation.

[0234] Configured as described above, the present disclosure enables the reduction of time-related costs when performing system design.

[0235] Note that at least one or more functions of the configuration collecting unit **131**, the feature value generating unit **132**, the distributed representation learning unit **133**, and the converting unit **134** may be performed by the information processing apparatus installed and connected anywhere on the network, that is, may be performed by so-called cloud computing.

[0236] Further, the abovementioned programs can be stored using various types of non-transitory computer-readable mediums and provided to a computer. The non-transitory computer-readable mediums include various types of tangible storage mediums. Examples of the non-transitory computer-readable mediums include a magnetic recording medium (e.g., floppy disk, magnetic tape, hard disk drive), a magneto-optical recording medium (e.g., magneto-optical disk), a CD-ROM (Read Only Memory), a CD-R, a CD-R/W, and a semiconductor memory (e.g., mask ROM, PROM (Programmable ROM), EPROM (Erasable PROM), flash ROM, and RAM (Random Access Memory)). The programs may also be provided to a computer by various types of transitory computer-readable mediums. Examples of the transitory computer-readable mediums include an electric signal, an optical signal, and an electromagnetic wave. The temporary computer-readable medium can provide the program to a computer via a wired communication path such as an electric wire or an optical fiber, or via a wireless communication path.

[0237] Although the present disclosure has been described above with reference to the example embodiments and so forth, the present disclosure is not limited to the above example embodiments. The configuration and details of the present disclosure can be changed in various manners that can be understood by one skilled in the art within the scope of the present disclosure. The above example embodiments can be combined with another example embodiment as required.

SUPPLEMENTARY NOTES

[0238] The whole or part of the example embodiments disclosed above can be described as in the supplementary notes. Below, the overview of the configurations of the information processing apparatus, the information processing method, and the program in the present disclosure will be described. However, the present disclosure is not limited to the following configurations.

Supplementary Note 1

[0239] An information processing apparatus comprising:

[0240] a configuration collecting unit that collects system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule;

[0241] a feature value generating unit that generates a feature value of the element in the system configuration information included by the design path information;

[0242] a distributed representation learning unit that acquires, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and

[0243] a converting unit that, for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generates design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

Supplementary Note 2

[0244] The information processing apparatus according to supplementary note 1, wherein the feature value generating unit generates property information representing a characteristic of the element as the feature value.

Supplementary Note 3

[0245] The information processing apparatus according to supplementary note 1, wherein the feature value generating unit generates type information representing a type of the element as the feature value.

Supplementary Note 4

[0246] The information processing apparatus according to supplementary note 1, wherein the feature value generating unit generates property information representing a characteristic of the element and type information representing a type of the element as the feature value.

Supplementary Note 5

[0247] The information processing apparatus according to supplementary note 4, wherein the feature value generating unit expresses the property information by an array indicating presence or absence of the characteristic of the element and also expresses the type information by an array indicating presence or absence of the type of the element, and connects the array of the property information and the array of the type information to generate as the feature value.

Supplementary Note 6

[0248] The information processing apparatus according to supplementary note 1, wherein when the feature value is associated with the element in the graph structure information and is input, the distributed representation learning unit acquires the parameter using a model that outputs the graph structure of the element accompanied by calculation of the distributed representation of the element.

Supplementary Note 7

[0249] The information processing apparatus according to supplementary note 6, wherein when the feature value is associated with the element in the graph structure information and is input, the distributed representation learning unit acquires the parameter by learning, using a preset correct value of an output, a model that outputs a predicted value of the graph structure of the element accompanied by calculation of the distributed representation of the element.

Supplementary Note 8

[0250] The information processing apparatus according to supplementary note 1, comprising:

[0251] a reward setting unit that, for each of the system configuration information included by the design path information with distributed representation, sets a reward for the system configuration information based on whether or not the element included by the system configuration information is a concrete element; and

[0252] a learning data generating unit that generates each of the system configuration information included by the design path information with distributed representation and the reward set for the system configuration information, as learning data to be machine-learned for generating a learning model that converts the system configuration information based on the concretization rule.

Supplementary Note 9

[0253] An information processing method by an information processing apparatus, the information processing method comprising:

[0254] collecting system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule; generating a feature value of the element in the system configuration information included by the design path information;

[0255] acquiring, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and

[0256] for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generating design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

Supplementary Note 10

[0257] A program comprising instructions for causing an information processing apparatus to execute processes to:

[0258] collect system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule;

[0259] generate a feature value of the element in the system configuration information included by the design path information;

[0260] acquire, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and

[0261] for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generate design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

DESCRIPTION OF REFERENCE NUMERALS

10	graph conversion device
11	configuration collecting unit
12	feature value generating unit
121	type identification information extracting unit
122	property information extracting unit
123	graph structure extracting unit
13	distributed representation learning unit
14	converting unit
20	storage device
30	input device
40	output device
50	automated system design learning apparatus
51	designing unit
52	reward setting unit
53	learning data generating unit

-continued

DESCRIPTION OF REFERENCE NUMERALS	
54	learning unit
60	storage device
70	input device
80	output device
100	information processing apparatus
101	CPU
102	ROM
103	RAM
104	programs
105	storage device
106	drive device
107	communication interface
108	input/output interface
109	bus
110	storage medium
111	communication network
131	configuration collecting unit
132	feature value generating unit
133	distributed representation learning unit
134	converting unit

1. An information processing apparatus comprising:

at least one memory storing processing instructions; and

at least one processor configured to execute the processing instructions to:

collect system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule;

generate a feature value of the element in the system configuration information included by the design path information;

acquire, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and

for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generate design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

2. The information processing apparatus according to claim **1**, wherein the at least one processor is configured to execute the processing instructions to

generate property information representing a characteristic of the element as the feature value.

3. The information processing apparatus according to claim **1**, wherein the at least one processor is configured to execute the processing instructions to

generate type information representing a type of the element as the feature value.

4. The information processing apparatus according to claim **1**, wherein the at least one processor is configured to execute the processing instructions to

generate property information representing a characteristic of the element and type information representing a type of the element as the feature value.

5. The information processing apparatus according to claim **4**, wherein the at least one processor is configured to execute the processing instructions to

express the property information by an array indicating presence or absence of the characteristic of the element and also express the type information by an array indicating presence or absence of the type of the element, and connect the array of the property information and the array of the type information to generate as the feature value.

6. The information processing apparatus according to claim **1**, wherein the at least one processor is configured to execute the processing instructions to

when the feature value is associated with the element in the graph structure information and is input, acquire the parameter using a model that outputs the graph structure of the element accompanied by calculation of the distributed representation of the element.

7. The information processing apparatus according to claim **6**, wherein the at least one processor is configured to execute the processing instructions to

when the feature value is associated with the element in the graph structure information and is input, acquire the parameter by learning, using a preset correct value of an output, a model that outputs a predicted value of the graph structure of the element accompanied by calculation of the distributed representation of the element.

8. The information processing apparatus according to claim **1**, wherein the at least one processor is configured to execute the processing instructions to:

for each of the system configuration information included by the design path information with distributed representation, set a reward for the system configuration information based on whether or not the element included by the system configuration information is a concrete element; and

generate each of the system configuration information included by the design path information with distributed representation and the reward set for the system configuration information, as learning data to be machine-learned for generating a learning model that converts the system configuration information based on the concretization rule.

9. An information processing method by an information processing apparatus, the information processing method comprising:

collecting system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule;

generating a feature value of the element in the system configuration information included by the design path information;

acquiring, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation

between the elements in the system configuration information by a graph structure and the feature value of the element; and

for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generating design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

10. A non-transitory computer-readable storage medium storing a program, the program comprising instructions for causing an information processing apparatus to execute processes to:

collect system configuration information included by design path information representing a process of designing the system configuration information, the system configuration information representing a system configuration including a combination of elements designed by converting an abstract element into a concrete element using a preset concretization rule;

generate a feature value of the element in the system configuration information included by the design path information;

acquire, by machine learning, a parameter for calculating a distributed representation of the element using graph structure information representing a relation between the elements in the system configuration information by a graph structure and the feature value of the element; and

for the design path information to be a target for learning by a learning model to be machine-learned so as to convert the abstract element into the concrete element using the concretization rule, generate design path information with distributed representation by calculating the distributed representation of the element in the system configuration information included by the design path information using the parameter and then assigning the distributed representation to the design path information.

* * * * *