(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 December 2016 (01.12.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/190841 A1

(51) International Patent Classification:
G06F 17/00 (2006.01) *G06F 21/00* (2006.01)

(21) International Application Number:
PCT/US2015/032212

(22) International Filing Date:
22 May 2015 (22.05.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **CHEN, Fei**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **VISWANATHAN, Krishnamurthy**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **GONZALEZ DIAZ, Maria Teresa**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US).

(74) Agents: **KIRCHEV, Ivan T.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

[Continued on next page]

(54) Title: MARGINAL DISTRIBUTION IN A GRAPHICAL MODEL

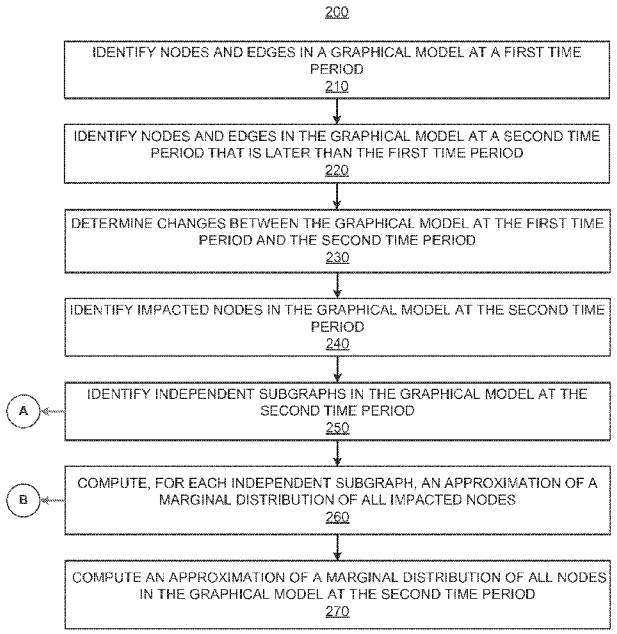

FIG. 2

(57) Abstract: An example method is provided in according with one implementation of the present disclosure. The method comprises identifying nodes and edges in a graphical model at a first time period, identifying nodes and edges in the graphical model at a second time period that is later than the first time period, determining changes between the graphical model at the first time period and the second time period, and identifying impacted nodes in the graphical model at the second time period. The method further comprises identifying independent subgraphs in the graphical model at the second time period, computing, for each independent subgraph, an approximation of a marginal distribution of all impacted nodes, and computing an approximation of a marginal distribution of all nodes in the graphical model at the second time period.

WO 2016/190841 A1

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

MARGINAL DISTRIBUTION IN A GRAPHICAL MODEL

BACKGROUND

[0001] Graphical models are graphs which compactly represent the joint distributions of many random variables. Graphical models have been widely used in various applications, such as malware detection, detecting malicious domains, topic modeling, information extraction, and many others.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Figure 1 is a schematic illustration of an example system for maintaining inference results on evolving graphical models in accordance with an implementation of the present disclosure.

[0003] Figure 2 illustrates a flowchart showing an example of a method for computing an approximation of a marginal distribution of all nodes in a graphical model in accordance with an implementation of the present disclosure.

[0004] Figure 3 is illustrates a flowchart showing an example of a method for identifying independent subgraphs in the graphical model in accordance with an implementation of the present disclosure.

[0005] Figure 4 illustrates a flowchart showing an example of a method for computing an approximation of the marginal distribution of all impacted nodes for each independent subgraph in accordance with an implementation of the present disclosure.

[0006] Figure 5 is an example block diagram illustrating a computer-readable medium in accordance with an implementation of the present disclosure.

DETAILED DESCRIPTION OF SPECIFIC EXAMPLES

[0007] As mentioned above, graphical models are widely used in many different applications. Graphical models may include a plurality of nodes or vertices connected by edges. In many of these applications, graphical models are constantly evolving. For example, in malware detection, the graphical model may be a bipartite graph where each node is a random binary variable which represents if a machine or a file is infected. There may be an edge between a file node and a machine node if the file is found on that machine. Such a graphical model changes

when new files are installed on the machines, files are deleted and machines are added/deleted from system.

[0008] Such evolving scenarios as well as many non-evolving cases may generally create a graph inference problem: given a graphical model that encodes a joint probability distribution, it may be difficult and time consuming to determine the marginal probability distribution of a particular node in the graph conditioned on the known value of some other nodes in the graph. As used herein, the term graph inference result refers to the marginal distribution of each node in the graph.

[0009] When the graphical model evolves, to keep the inference results (e.g., the marginal distribution of each random node in the graph) up to date, one approach is to run graphical inference algorithms (e.g., belief propagation, Gibbs sampling, etc.) from scratch on the entire graph. However, on large scale graphs (i.e., graphs containing millions of vertices), such an approach can often takes hours to finish. For some areas (e.g., such as security applications), which need real-time or near real-time response, rerunning-from-scratch is not a viable approach.

[0010] In this regard, according to examples, techniques for computing or maintaining inference results on evolving graphical models are described herein. In one example, changes in the graphical model may be identified between snapshots of the model in two different time periods. The proposed techniques efficiently identify relatively small subgraphs in the changed large graphical models. Then, the techniques run an inference algorithm on these subgraphs to compute an approximation of marginal distribution of all impacted nodes in the subgraphs as fast as possible. Thus, the techniques described herein perform graph inference incrementally (e.g., on a small neighborhood of impacted nodes), and not on the entire graphical model. For example, if a change happens to a first node far away from a second node, the changes for the second node may be approximately calculated because the changed first node would not have much effect over the second node. Therefore, when the changes between a graphical model in two different times are small, the proposed techniques offer a faster way to re-run an inference algorithm on the entire graph from scratch.

[0011] In some examples, when there are small changes in a graphical model, the inference results may not change dramatically for most of the nodes.

Therefore, the techniques described herein propose to first identify impacted subgraphs of the original graph that are likely to contain nodes whose marginal distribution may change significantly based on the changes in the graph between the two different time periods. Then, the techniques propose to only run the inference algorithm on those subgraphs before combining data for the subgraphs with the data for the rest of the graph.

[0012] In one example, a processor may identify nodes and edges in a graphical model at a first time period and at a second time period that is later than the first time period. The processor may further: determine changes between the graphical model at the first time period and the second time period, identify impacted nodes in the graphical model at the second time period, and identify independent subgraphs in the graphical model at the second time period. Finally, the processor may compute an approximation of a marginal distribution of all impacted nodes for each independent subgraph, and may compute an approximation of a marginal distribution of all nodes in the graphical model at the second time period.

[0013] Thus, the proposed techniques may consider any type of change in a graphical model (e.g., deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed, etc.) and may apply any inference algorithm to the determined subgraphs. Further, the described techniques may efficiently compute inference results on evolving power-law graphs that include high-degree nodes.

[0014] In the following detailed description, reference is made to the accompanying drawings, which form a part hereof, and in which is shown by way of illustration specific examples in which the disclosed subject matter may be practiced. It is to be understood that other examples may be utilized and structural or logical changes may be made without departing from the scope of the present disclosure. The following detailed description, therefore, is not to be taken in a limiting sense, and the scope of the present disclosure is defined by the appended claims. Also, it is to be understood that the phraseology and terminology used herein is for the purpose of description and should not be regarded as limiting. The use of “including,” “comprising” or “having” and variations thereof herein is meant to encompass the items listed thereafter and equivalents thereof as well as

additional items. Furthermore, the term "based on," as used herein, means "based at least in part on." It should also be noted that a plurality of hardware and software based devices, as well as a plurality of different structural components may be used to implement the disclosed methods and devices.

[0015] Referring now to the figures, Figure 1 is a schematic illustration of an example system 10 for maintaining inference results on evolving graphical models. As noted above, the term "inference result" refers to an approximation of a marginal distribution of all nodes in the graphical model. The illustrated system 10 is capable of carrying out the techniques described below. As shown in Figure 1, the system 10 is depicted as including at least one a computing device 100. In the embodiment of Figure 1, computing device 100 includes a processor 102, an interface 106, and a machine-readable storage medium 110. Although only computing device 100 is described in details below, the techniques described herein may be performed with several computing devices or by engines distributed on different devices.

[0016] The computing device 100 may be any type of a computing device and may include at least engines 120-160. Engines 120-150 may or may not be part of the machine-readable storage medium 110. In one example, the computing device 100 may be an independent computing device. In another alternative example, engines 120-160 may be distributed between the computing device 100 and others computing devices. The computing device 100 may include additional components and some of the components depicted therein may be removed and/or modified without departing from a scope of the system that allows for carrying out the functionality described herein. It is to be understood that the operations described as being performed by the engines 120-160 of the computing device 100 that are related to this description may, in some implementations, be performed by external engines (not shown) or distributed between the engines of the computing device 100 and other electronic/computing devices.

[0017] Processor 102 may be central processing unit(s) (CPUs), microprocessor(s), and/or other hardware device(s) suitable for retrieval and execution of instructions (not shown) stored in machine-readable storage medium 110. Processor 102 may fetch, decode, and execute instructions to identify different groups in a dataset. As an alternative or in addition to retrieving and

executing instructions, processor 102 may include electronic circuits comprising a number of electronic components for performing the functionality of instructions.

[0018] Interface 106 may include a number of electronic components for communicating with various devices. For example, interface 106 may be an Ethernet interface, a Universal Serial Bus (USB) interface, an IEEE 1394 (Firewire) interface, an external Serial Advanced Technology Attachment (eSATA) interface, or any other physical connection interface suitable for communication with the computing device. Alternatively, interface 106 may be a wireless interface, such as a wireless local area network (WLAN) interface or a near-field communication (NFC) interface that is used to connect with other devices/systems and/or to a network. In one example, the network may be a mesh sensor network (not shown). The network may include any suitable type or configuration of network to allow for communication between the computing device 100, and any other devices/systems (e.g., other computing devices, displays, etc.), for example, to send and receive data to and from a corresponding interface of another device.

[0019] Each of the engines 120-160 may include, for example, at least one hardware device including electronic circuitry for implementing the functionality described below, such as control logic and/or memory. In addition or as an alternative, the engines 120-160 may be implemented as any combination of hardware and software to implement the functionalities of the engines. For example, the hardware may be a processor and the software may be a series of instructions or microcode encoded on a machine-readable storage medium and executable by the processor. Therefore, as used herein, an engine may include program code, (e.g., computer executable instructions), hardware, firmware, and/or logic, or combination thereof to perform particular actions, tasks, and functions described in more detail herein in reference to Figures 2-5.

[0020] In one example, the identification engine 120 may identify nodes and edges in a graphical model at a first time period and at a second time period that is later than the first time period. In some implementations, the graphical model may be represented as a first snapshot of the graphical model at the first time period and a second snapshot of the graphical model at the second time period.

[0021] The analysis engine 130 may determine changes between the graphical model at the first time period and the second time period, and may

identify impacted nodes in the graphical model at the second time period. In one example, the changes between the graphical model at the first time period and the second time period include at least one of: deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed. It may also include changes to factors on the edges and cliques (these factors encode the joint probability distribution of the nodes in the graphical model). In other examples, other types of changes may be detected. In addition, each of the impacted nodes may have a topology change from the first time period to the second time period. The topology change may include adding or deleting an edge of the node, having a brand new node, etc.

[0022] The subgraphs engine 140 may identify independent subgraphs in the graphical model at the second time period. In one example, the subgraphs engine 130 may: identify evidence nodes that have changed between the first time period and the second time period, identify high-degree nodes, identify active nodes, remove all edges that are not connected to active nodes, and determine a plurality of independent subgraphs. As used herein, the term "active node" refers to a node that has at least one active edge or has no inactive edges. As used herein, the term "evidence node" refers to a node in the graphical model that has a known value. Further, as used herein, the term "high-degree node" refers to a node that is connected to a very large number of other nodes and the factors on the edges are such that a change in its neighbor nodes will not dramatically change its distribution.

[0023] The inference engine 150 may compute an approximation of a marginal distribution of all impacted nodes in each of the independent subgraphs. Thus, the inference engine 150 may apply an inference algorithm to each independent subgraph that was affected by the change in the graphical model.

[0024] The graphical model engine 160 may compute an approximation of a marginal distribution of all nodes in the graphical model at the second time period. Thus, the graphical model engine 160 may maintain the inference results of an evolving graphical model at a much faster rate.

[0025] Figure 2 illustrates a flowchart showing an example of a method 200 for computing an approximation of a marginal distribution of all nodes in a graphical model (i.e., for maintaining inference results on evolving graphical models).

Although execution of the method 200 is described below with reference to the system 10, the components for executing the method 200 may be spread among multiple devices/systems. The method 200 may be implemented in the form of executable instructions stored on a machine-readable storage medium, and/or in the form of electronic circuitry.

[0026] In one example, the method 200 can be executed by at least one processor of a computing device (e.g., processor 102 of device 100). In other examples, the method may be executed by another processor in communication with the system 10. Various elements or blocks described herein with respect to the method 200 are capable of being executed simultaneously, in parallel, or in an order that differs from the illustrated serial manner of execution. The method 200 is also capable of being executed using additional or fewer elements than are shown in the illustrated examples.

[0027] The method 200 begins at 210, where at least one processor may identify nodes and edges in a graphical model at a first time period. For example, the graphical model may be represented as a first snapshot of the graphical model at the first time period. At 220, the processor may identify nodes and edges in the graphical model at a second time period that is later than the first time period. The graphical model may be represented as a second snapshot of the graphical model at the second time period. In one example, the processor may receive and/or identify the following information: a first graph snapshot G_t for the first time period t, its set of nodes or vertices V_t and edges E_t, a second graph snapshot G_{t+1} for the first time period $t+1$, its set of nodes or vertices V_{t+1}, and edges E_{t+1}. Thus, a change or evolution in the graphical model is examined by the processor. The processor may use different techniques to identify nodes and edges in the graphical model at the different time periods.

[0028] At 230, the processor may determine changes between the graphical model at the first time period t and the second time period t+1. For example, the changes between the graphical model at the first time period and the second time period include at least one of: deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed. In other implementations, other changes in the graphical model may also be identified. The

processor may use various techniques to determine the changes in the graphical model.

[0029] Next, the processor may identify impacted nodes *I* in the graphical model at the second time period (at 240). For example, the processor may use the changes in the graphical model determined in block 230 to identify the impacted nodes *I* in the graph. In one implementation, each of the impacted nodes has a topology change from the first time period to the second time period. The topology change may include adding or deleting an edge of the node, having a brand new node, etc. As noted below, the values of the impacted nodes may be recomputed using an inference to calculate an approximation of the marginal distribution of all nodes.

[0030] At 250, the processor may identify independent subgraphs C_{t+1} in the graphical model at the second time period. When the changes in the graphical model between the first and the second time periods are small, the inference results may not change dramatically for most of the nodes. Thus, the processor may first identify the impacted regions (i.e., subgraphs) within the original large graph that are likely to contain nodes whose marginal distribution may change significantly based on the changes in the graph between the two different time periods.

[0031] With continued reference to Figure 2, the processor may compute, for each independent subgraph, an approximation of the marginal distribution of all impacted nodes (at 260). The process for computing an approximation of the marginal distribution of all impacted nodes for each independent subgraph is described in below in relation to Figure 4. As described in additional details below, the processor applies an inference algorithm *A* to each subgraph to compute an approximation of the marginal distribution of all impacted nodes in each of the subgraphs. Therefore, the processor applies the inference algorithm only to the identified impacted subgraphs and not to the entire graphical model. Considering the size of many graphical models, this selective process speeds up the generation of inference results.

[0032] At 270, the processor may compute an approximation $\tilde{A}(G_{t+1})$ of the marginal distribution of all nodes in the graphical model $A(G_{t+1})$ at the second time

period. In other words, the processor returns the approximated marginal distribution $\tilde{A}(G_{t+1})$ for the entire graphical model at the second period of time *t+1* in order to maintain the inference results on the evolving graphical model in an expedited paste. In one example, the processor may merge the calculated approximation of the marginal distribution for all subgraphs A(I) with the unchanged nodes at time *t+1* in the graphical model $A(G_t, I)$: $\tilde{A}(G_{t+1}) \leftarrow A(I) \cup A(G_t - I)$.

[0033] A method 300 for identifying independent subgraphs in the graphical model at the second time period is described in more details in relation to Figure 3. The method 300 can be executed by at least one processor of a computing device (e.g., processor 102 of device 100). In other examples, the method may be executed by another processor in communication with the system 10. The trained model may be generated by using a training dataset.

[0034] The method 300 begins at 310, where the processor may identify evidence nodes E that have changed between the first time period and the second time period. As noted above "evidence node" refers to a node in the graphical model that has a known value and, therefore, there is no randomness in that respect. Thus, their distribution is unaffected by the value of any other nodes. For an evidence node, changes in its few neighbors will not dramatically change the distribution of the node. In other words, the processor may identify evidence nodes E that have not changed between the first time period *t* and the second time period *t*+1 in the group of all nodes $V_{t+1} \cap V_t$. Various techniques may be used to identify evidence nodes that have changed between the two time periods.

[0035] At 320, the processor may identify high-degree nodes in the graphical model. A high-degree node is a node that is connected to a very large number of other nodes such that a change in its neighbor nodes will not dramatically change its distribution. In some examples, the graphical models include power law graphs. In such graphs, some high-degree nodes have thousands of neighbors. In one implementation, the processor may use a predetermined threshold *d* of high-degree nodes to identify the high-degree nodes in the group of all nodes $V_{t+1} \cap V_t$. Changes in high-degree nodes will not dramatically change the distribution of these high-degree nodes. Further, high-degree nodes also block the changes or dependencies to other neighbors. Thus, high-degree nodes and the evidence

nodes may block the dependency among nodes in the graph. Any path that contains either an evidence node or a high-degree node may be identified as an inactive path. Thus, the processor may identify the union *H* of high-degree nodes and relevance nodes.

[0036] At 330, the processor may identify active nodes W_{t+1} in the graphical model at the second time period. As noted above, an "active node" refers to a node that has at least one active edge or has no inactive edges. In other words, the processor may identify active nodes W_{t+1} in all nodes V_{t+1} at the second time period, where the active nodes are not high-degree nodes or evidence nodes: $W_{t+1} \leftarrow V_{t+1} - (E \cup H)$. One technique that may be used to identify the active nodes is to identify active paths between the nodes by using dependent separation. For example, two nodes X and Y are conditionally independent given Z if knowledge about X gives no extra information about Y once there is knowledge of Z. In other words, once there is knowledge of Z, X adds nothing to the knowledge about Y. Thus, a path between two nodes is active if it carries information or dependence. Two nodes X and Y might be connected by many paths in a graph, where all, some, or none of the paths are active. Therefore, X and Y are dependent-separated if all the paths that connect them are inactive, or, equivalently, if no path between them is active.

[0037] At 340, the processor may remove all edges E'_{t+1} that are not connected to active nodes and may determine a plurality of independent subgraphs C_{t+1} in the graphical model G', where $G'=< V_{t+1}, E'_{t+1}>$ (at 350). Thus, given the evidence in the graphical model (e.g. those nodes whose assignments are known, the dependent separation between the nodes, etc.), the processor may find all active paths, and that information may be used to identify the active nodes and to break the graph into several smaller independent subgraphs if there is no active path connecting them. In other words, a first subgraph and a second subgraph of the graphical model are independent when there are no active paths connecting the subgraphs, and where changes in the first independent subgraph do not affect the marginal distribution of any nodes in the second independent subgraph.

[0038] Figure 4 illustrates a flowchart showing an example of a method 400 for computing an approximation of the marginal distribution of all impacted nodes

for each independent subgraph. The method 400 can be executed by at least one processor of a computing device (e.g., processor 102 of device 100). In other examples, the method may be executed by another processor in communication with the system 10.

[0039] The method 400 begins at 410 where the processor may identify each impacted node in the subgraph. Then, at 420, the processor may identify all k-neighborhood nodes for each impacted node in the subgraph. As used herein, the term "k-neighborhood nodes" refers to nodes that are reachable from a node and are within a predetermine distance *k*. The size of the *k*-neighborhood may vary and may be fixed or flexible.

[0040] In one implementation, the processor may use the length of a path between two nodes (i.e., the number of other nodes it needs to go through to get from one node to the other) to identify all k-neighborhood nodes for each impacted node in the subgraph. For example, even if nodes X and Y are connected via some active paths, if all these active paths are very long, the dependence between X and Y is very weak. Therefore, changes in X's distribution only impact Y's distribution slightly. The processor, therefore, identifies the k-neighborhood of nodes to be considered for a subgraph. For instance, a threshold (e.g., Z number of nodes to get from node X to node Y) that would identify long paths may be used.

[0041] At 430, the processor may add the identified k-neighborhood nodes for each impacted node to the set of impacted nodes. In other words, the processor "grows" the set of impacted nodes in the subgraph by adding non-impacted nodes to identify the right size subgraph to which to apply an inference algorithm. The right size subgraph is a subgraph where changes in the nodes would strongly impact the marginal distribution of adjacent nodes.

[0042] At 440, the processor may apply an inference algorithm *A* to each independent subgraph. In other words, the inference algorithm is applied to the entire independent subgraph – including impacted and non-impacted nodes. That way, the processor may receive inference results for all nodes within the independent subgraph.

[0043] Figure 5 illustrates a computer 501 and a non-transitory machine-readable medium 505 according to an example. In one example, the computer 501 maybe similar to the computing device 100 of the system 10 or may include a

plurality of computers. For example, the computer may be a server computes, a workstation computer, a desktop computer, a laptop, a mobile device, or the like, and may be part of a distributed system. The computer may include one or more processors and one or more machine-readable storage media. In one example, the computer may include a user interface (e.g., touch interface, mouse, keyboard, gesture input device, etc.).

[0044] Computer 501 may perform methods 200-400 and variations thereof. Additionally, the functionality implemented by computer 501 may be part of a larger software platform, system, application, or the like. Computer 501 may be connected to a database (not shown) via a network. The network may be any type of communications network, including, but not limited to, wire-based networks (e.g., cable), wireless networks (e.g., cellular, satellite), cellular telecommunications network(s), and IP-based telecommunications network(s) (e.g., Voice over Internet Protocol networks). The network may also include traditional landline or a public switched telephone network (PSTN), or combinations of the foregoing.

[0045] The computer 501 may include a processor 503 and non-transitory machine-readable storage medium 505. The processor 503 (e.g., a central processing unit, a group of distributed processors, a microprocessor, a microcontroller, an application-specific integrated circuit (ASIC), a graphics processor, a multiprocessor, a virtual processor, a cloud processing system, or another suitable controller or programmable device) and the storage medium 305 may be operatively coupled to a bus. Processor 503 can include single or multiple cores on a chip, multiple cores across multiple chips, multiple cores across multiple devices, or combinations thereof.

[0046] The storage medium 505 may include any suitable type, number, and configuration of volatile or non- volatile machine-readable storage media to store instructions and data. Examples of machine-readable storage media in include read-only memory ("ROM"), random access memory ("RAM") (e.g., dynamic RAM ["DRAM"], synchronous DRAM ["SDRAM"], etc.), electrically erasable programmable read-only memory ("EEPROM"), magnetoresistive random access memory (MRAM), memristor, flash memory, SD card, floppy disk, compact disc read only memory (CD-ROM), digital video disc read only memory (DVD-ROM),

and other suitable magnetic, optical, physical, or electronic memory on which software may be stored.

[0047] Software stored on the non-transitory machine-readable storage media 505 and executed by the processor 503 includes, for example, firmware, applications, program data, filters, rules, program modules, and other executable instructions. The processor 503 retrieves from the machine-readable storage media 505 and executes, among other things, instructions related to the control processes and methods described herein.

[0048] The processor 503 may fetch, decode, and execute instructions 307-311 among others, to implement various processing. As an alternative or in addition to retrieving and executing instructions, processor 503 may include at least one integrated circuit (IC), other control logic, other electronic circuits, or combinations thereof that include a number of electronic components for performing the functionality of instructions 507-515. Accordingly, processor 503 may be implemented across multiple processing units and instructions 507-515 may be implemented by different processing units in different areas of computer 501.

[0049] The instructions 507-515 when executed by processor 503 (e.g., via one processing element or multiple processing elements of the processor) can cause processor 503 to perform processes, for example, methods 200-400, and/or variations and portions thereof. In other examples, the execution of these and other methods may be distributed between the processor 503 and other processors in communication with the processors 603.

[0050] For example, data identification instructions 507 may cause processor 503 to identify nodes and edges in a graphical model at a first time period and at a second time period that is later than the first time period. In some implementations, the graphical model may be represented as a first snapshot of the graphical model at the first time period and a second snapshot of the graphical model at the second time period. These instructions may function similarly to the techniques described in blocks 210 and 220 of method 200.

[0051] Analysis instructions 509 may cause the processor 503 to determine changes between the graphical model at the first time period and the second time period, and to identify impacted nodes in each of the independent subgraphs in the

graphical model at the second time period. Each of the impacted nodes may have a topology change from the first time period to the second time period. In some examples, each of the impacted nodes has a value in the second time period that is different from the value of the node in the first time period. These instructions may function similarly to the techniques described in blocks 230-240 of method 200. In one example, the changes between the graphical model at the first time period and the second time period include at least one of: deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed. In other example, other types of changes may be detected.

[0052] Subgraphs instructions 511 may cause the processor 503 to identify independent subgraphs in the graphical model at the second time period. These instructions may function similarly to the techniques described in block 250 of method 200 and to the techniques described in method 300. For example, the subgraphs instructions 511 may cause the processor 503 to: identify evidence nodes that have changed between the first time period and the second time period, identify high-degree nodes, identify active nodes, remove all edges that are not connected to active nodes, and determine a plurality of independent subgraphs.

[0053] Inference instructions 513 may cause the processor 503 to compute an approximation of a marginal distribution of all impacted nodes in each of the independent subgraphs. These instructions may function similarly to the techniques described blocks 260 of method 200 and to the techniques described in method 400. Thus, inference instructions 513 may cause the processor 503 to apply an inference algorithm to each independent subgraph that was affected by the change in the graphical model.

[0054] Graphical model instructions 515 may cause the processor 503 to compute an approximation of a marginal distribution of all nodes in the graphical model at the second time period. These instructions may function similarly to the techniques described blocks 270 of method 200. Thus, graphical model instructions 515 may cause the processor 503 to maintain the inference results of an evolving graphical model at a very fast paste.

[0055] In the foregoing description, numerous details are set forth to provide an understanding of the subject matter disclosed herein. However, implementations may be practiced without some or all of these details. Other

implementations may include modifications and variations from the details discussed above. It is intended that the appended claims cover such modifications and variations.

What is claimed is:

1. A method comprising, by at least one processor:

identifying nodes and edges in a graphical model at a first time period;

identifying nodes and edges in the graphical model at a second time period that is later than the first time period;

determining changes between the graphical model at the first time period and the second time period;

identifying impacted nodes in the graphical model at the second time period;

identifying independent subgraphs in the graphical model at the second time period;

computing, for each independent subgraph, an approximation of a marginal distribution of all impacted nodes; and

computing an approximation of a marginal distribution of all nodes in the graphical model at the second time period.

2. The method of claim 1, further comprising, by at least one processor:

identifying evidence nodes that have changed between the first time period and the second time period;

identifying high-degree nodes;

identifying active nodes;

removing all edges that are not connected to active nodes; and

determining a plurality of independent subgraphs.

3. The method of claim 1, further comprising, for each independent subgraph, by at least one processor:

identifying each impacted node in the subgraph;

identifying all k-neighborhood nodes for each impacted node in the subgraph;

adding the identified k-neighborhood nodes for each impacted node to the set of impacted nodes; and

applying an inference algorithm to each independent subgraph.

4. The method of claim 1, wherein changes between the graphical model at the first time period and the second time period include at least one of: deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed.

5. The method of claim 1, wherein each of the impacted nodes has a topology change from the first time period to the second time period.

6. The method of claim 1, wherein a first subgraph and a second subgraph of the graphical model are independent when there are no active paths connecting the subgraphs, and wherein changes in the first independent subgraph do not affect the marginal distribution of any nodes in the second independent subgraph.

7. The method of claim 1, wherein the graphical model is represented as a first snapshot of the graphical model at the first time period and a second snapshot of the graphical model at the second time period.

8. A system comprising:

an identification engine to identify nodes and edges in a graphical model at a first time period and at a second time period that is later than the first time period, wherein the graphical model is represented as a first snapshot of the graphical model at the first time period and a second snapshot of the graphical model at the second time period;

an analysis engine to:

determine changes between the graphical model at the first time period and the second time period, and

identify impacted nodes in the graphical model at the second time period;

a subgraphs engine to identify independent subgraphs in the graphical model at the second time period;

an inference engine to compute an approximation of a marginal distribution of all impacted nodes in each of the independent subgraphs; and

a graphical model engine to compute an approximation of a marginal distribution of all nodes in the graphical model at the second time period.

9. The system of claim 8, wherein for each independent subgraph the subgraphs engine is further to:

identify evidence nodes that have changed between the first time period and the second time period;

identify high-degree nodes;

identify active nodes;

remove all edges that are not connected to active nodes; and

determine a plurality of independent subgraphs.

10. The system of claim 8, wherein the inference engine is further to:

identify each impacted node in the subgraph;

identify all k-neighborhood nodes for each impacted node in the subgraph;

add the identified k-neighborhood nodes for each impacted node to the set of impacted nodes; and

apply an inference algorithm to each independent subgraph.

11. The system of claim 8, wherein changes between the graphical model at the first time period and the second time period include at least one of: deleted nodes, inserted nodes, nodes with changed edges, and nodes whose previously unknown values are revealed.

12. A non-transitory machine-readable storage medium encoded with instructions executable by at least one processor, the machine-readable storage medium comprising instructions to:

identify nodes and edges in a graphical model at a first time period;

identify nodes and edges in the graphical model at a second time period that is later than the first time period;

determine changes between the graphical model at the first time period and the second time period;

identify impacted nodes in the graphical model at the second time period, wherein each of the impacted nodes has a topology change from the first time period to the second time period;

identify independent subgraphs in the graphical model at the second time period;

compute an approximation of a marginal distribution of all impacted nodes in each of the independent subgraphs; and

compute an approximation of a marginal distribution of all nodes in the graphical model at the second time period.

13. The non-transitory machine-readable storage medium of claim 12, further comprising instructions to:

identify evidence nodes that have changed between the first time period and the second time period;

identify high-degree nodes;

identify active nodes;

remove all edges that are not connected to active nodes; and

determine a plurality of independent subgraphs.

14. The non-transitory machine-readable storage medium of claim 13, further comprising instructions to:

identify each impacted node in the subgraph;

identify all k-neighborhood nodes for each impacted node in the subgraph;

add the identified k-neighborhood nodes for each impacted node to the set of impacted nodes; and

apply an inference algorithm to each independent subgraph.

15. The non-transitory machine-readable storage medium of claim 12, wherein a first subgraph and a second subgraph of the graphical model are independent when there are no active paths connecting the subgraphs, and wherein changes in the first independent subgraph do not affect the marginal distribution of any nodes in the second independent subgraph.

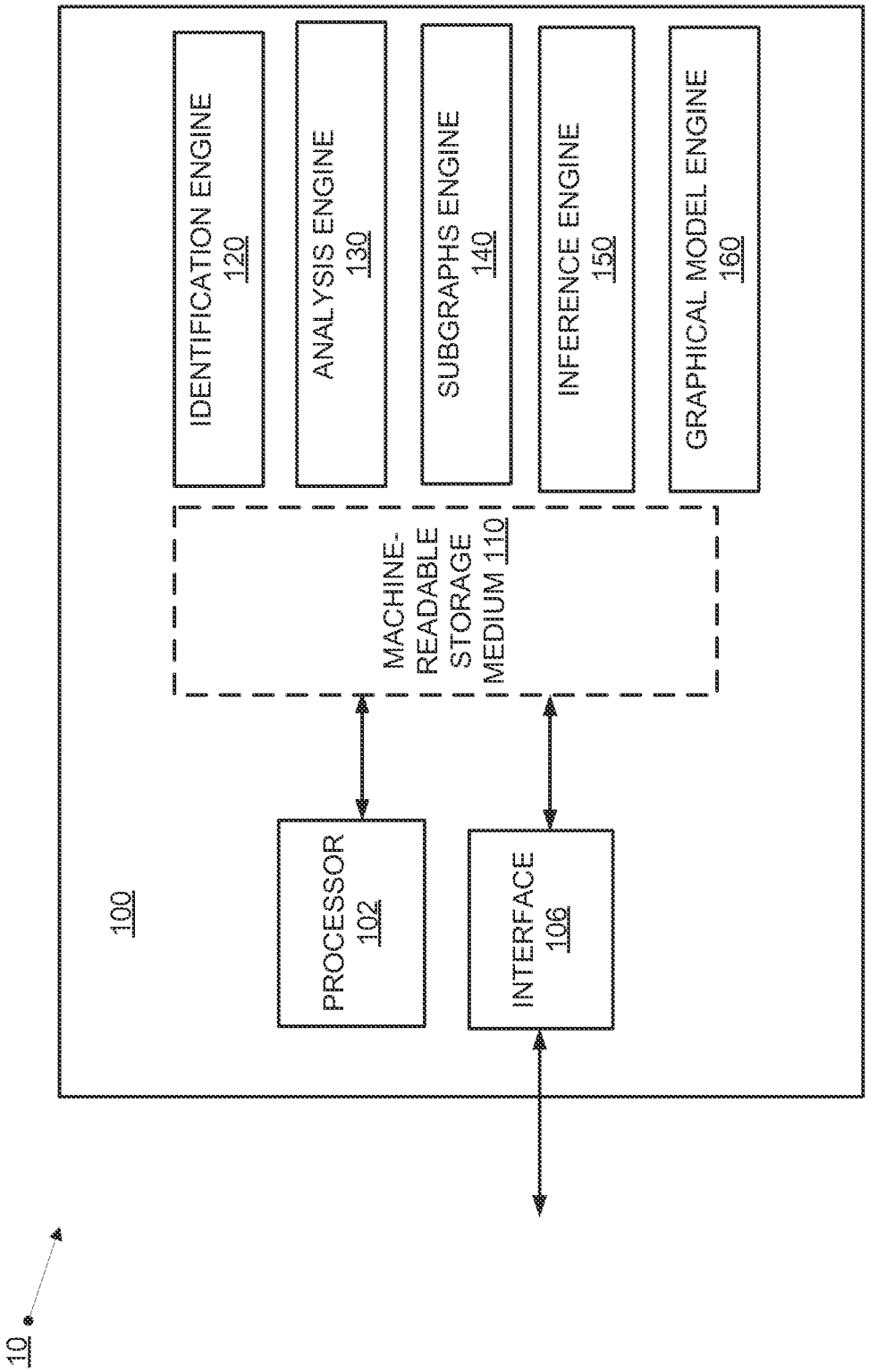
10
100
IDENTIFICATION ENGINE 120
ANALYSIS ENGINE 130
SUBGRAPHS ENGINE 140
INFERENCE ENGINE 150
GRAPHICAL MODEL ENGINE 160
MACHINE-READABLE STORAGE MEDIUM 110
PROCESSOR 102
INTERFACE 106

FIG. 1

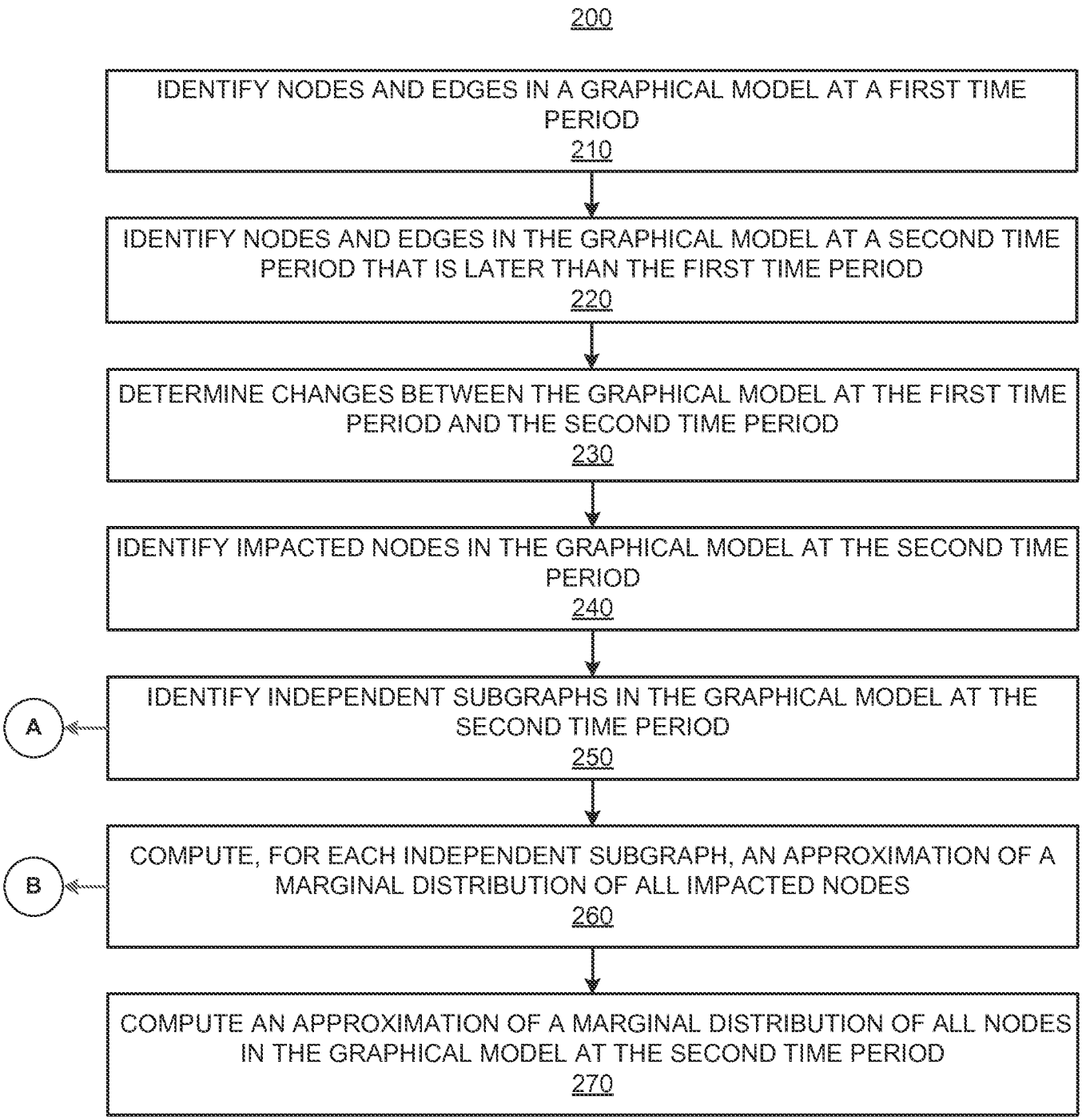
200
IDENTIFY NODES AND EDGES IN A GRAPHICAL MODEL AT A FIRST TIME PERIOD
210
IDENTIFY NODES AND EDGES IN THE GRAPHICAL MODEL AT A SECOND TIME PERIOD THAT IS LATER THAN THE FIRST TIME PERIOD
220
DETERMINE CHANGES BETWEEN THE GRAPHICAL MODEL AT THE FIRST TIME PERIOD AND THE SECOND TIME PERIOD
230
IDENTIFY IMPACTED NODES IN THE GRAPHICAL MODEL AT THE SECOND TIME PERIOD
240
A
IDENTIFY INDEPENDENT SUBGRAPHS IN THE GRAPHICAL MODEL AT THE SECOND TIME PERIOD
250
B
COMPUTE, FOR EACH INDEPENDENT SUBGRAPH, AN APPROXIMATION OF A MARGINAL DISTRIBUTION OF ALL IMPACTED NODES
260
COMPUTE AN APPROXIMATION OF A MARGINAL DISTRIBUTION OF ALL NODES IN THE GRAPHICAL MODEL AT THE SECOND TIME PERIOD
270

FIG. 2

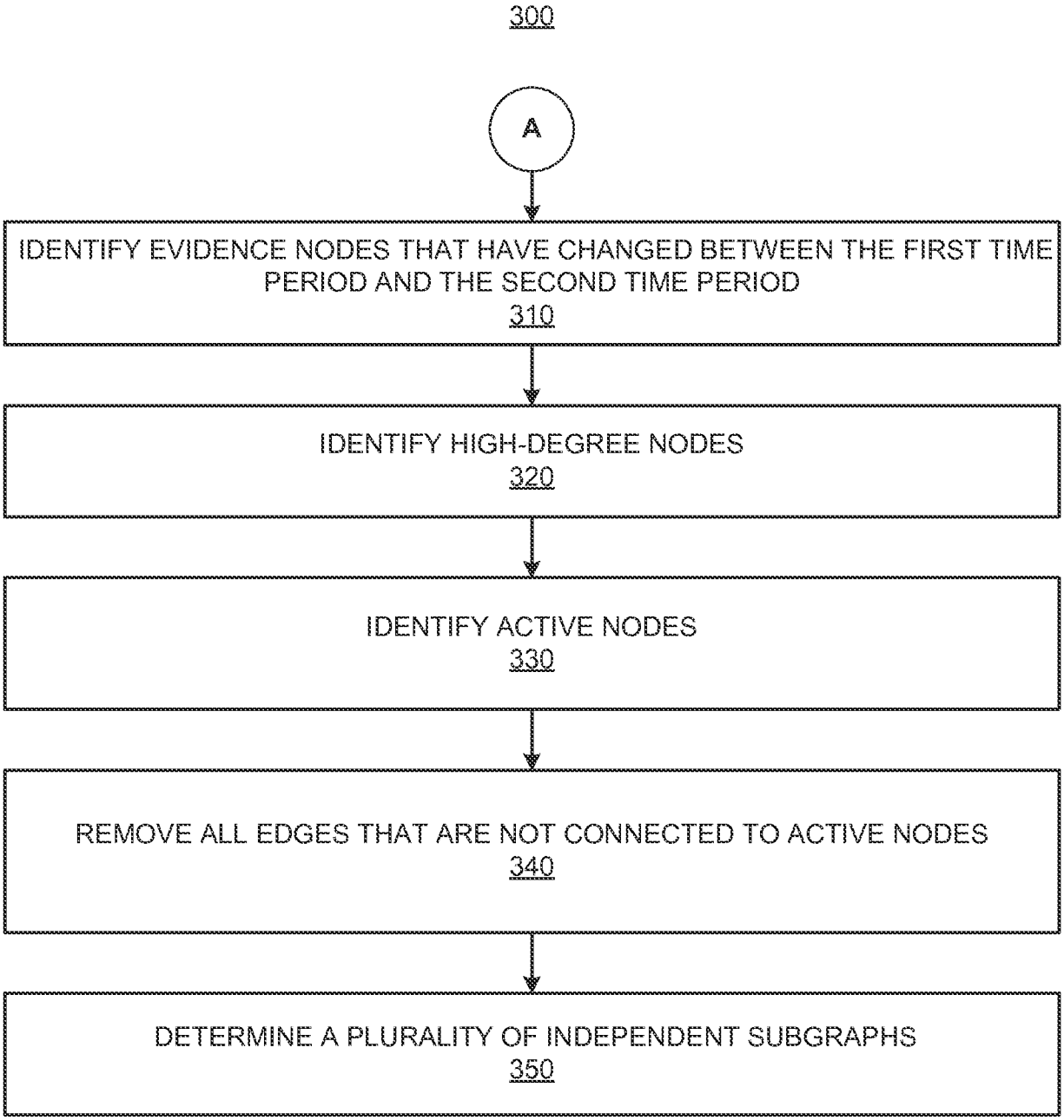
300
A
IDENTIFY EVIDENCE NODES THAT HAVE CHANGED BETWEEN THE FIRST TIME PERIOD AND THE SECOND TIME PERIOD
310
IDENTIFY HIGH-DEGREE NODES
320
IDENTIFY ACTIVE NODES
330
REMOVE ALL EDGES THAT ARE NOT CONNECTED TO ACTIVE NODES
340
DETERMINE A PLURALITY OF INDEPENDENT SUBGRAPHS
350

FIG. 3

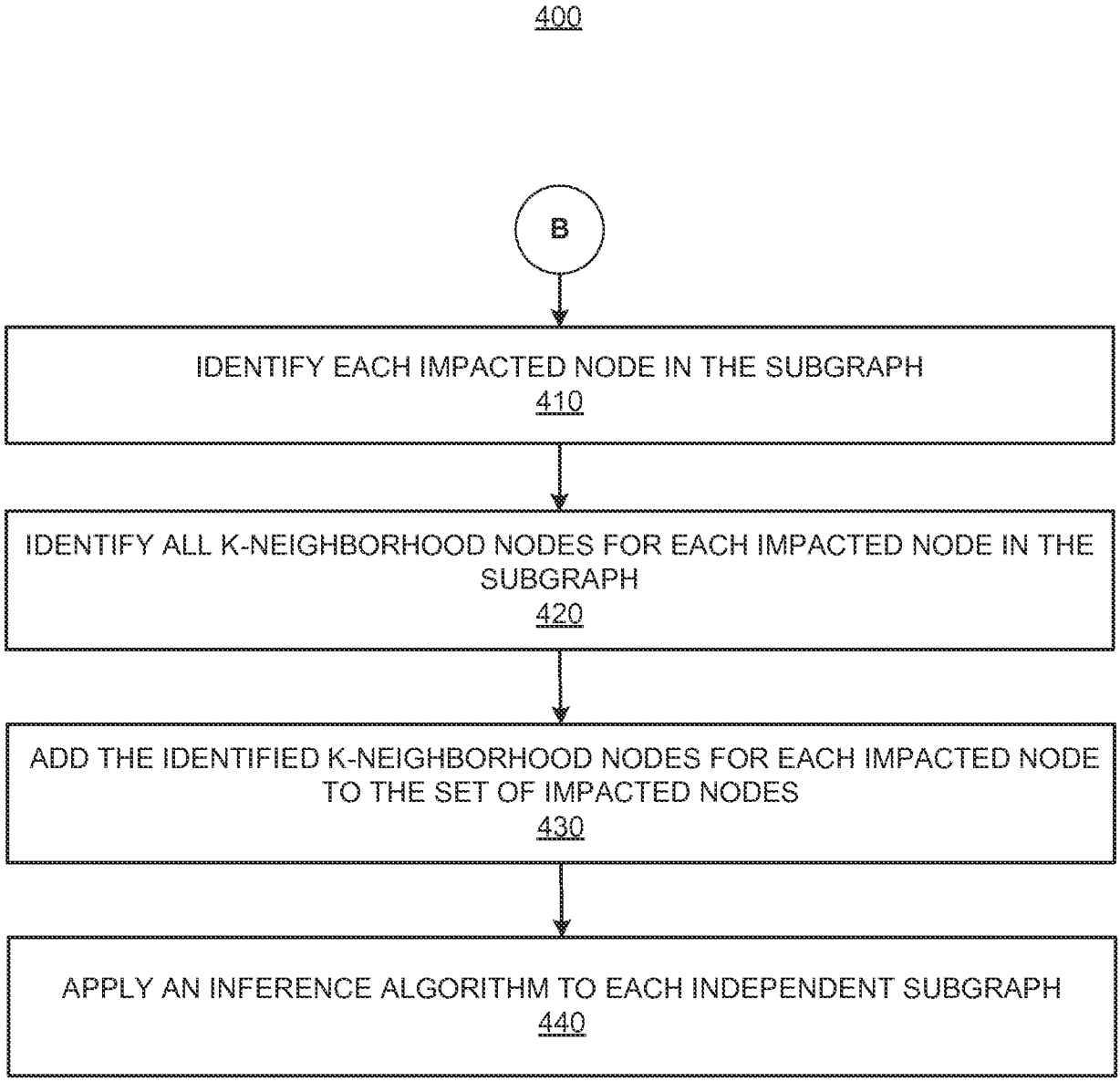
400
B
IDENTIFY EACH IMPACTED NODE IN THE SUBGRAPH
410
IDENTIFY ALL K-NEIGHBORHOOD NODES FOR EACH IMPACTED NODE IN THE SUBGRAPH
420
ADD THE IDENTIFIED K-NEIGHBORHOOD NODES FOR EACH IMPACTED NODE TO THE SET OF IMPACTED NODES
430
APPLY AN INFERENCE ALGORITHM TO EACH INDEPENDENT SUBGRAPH
440

FIG. 4

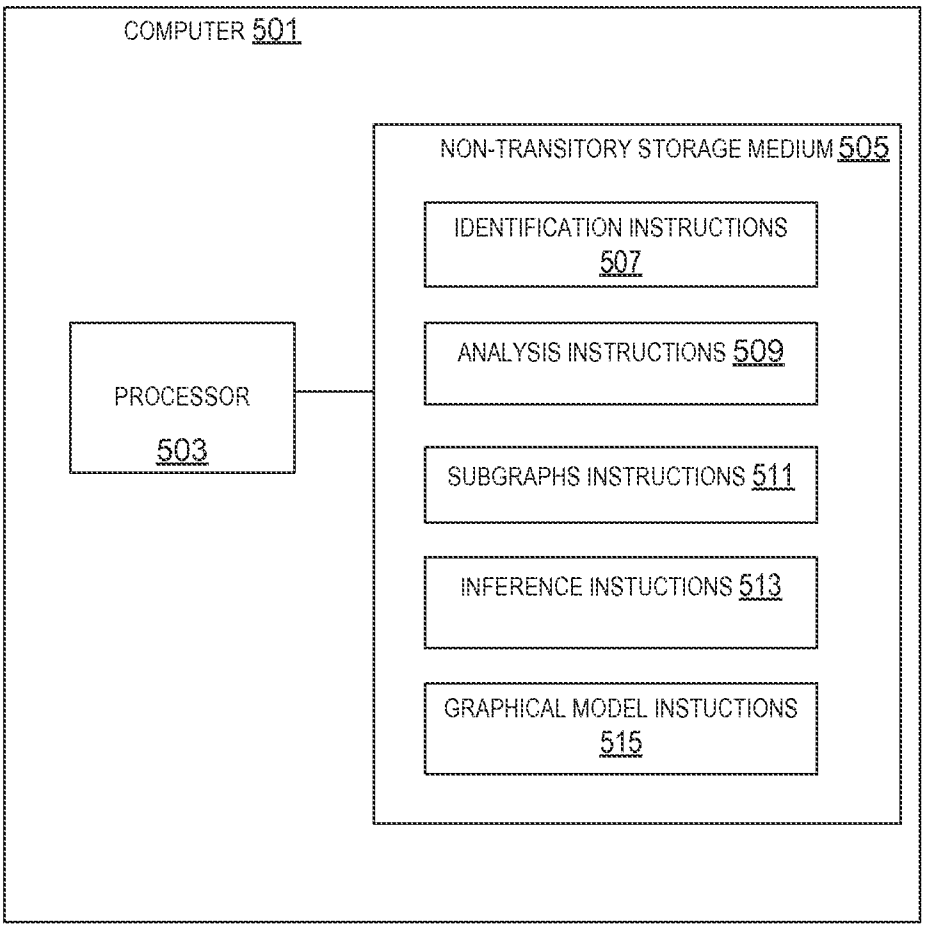
COMPUTER 501
NON-TRANSITORY STORAGE MEDIUM 505
IDENTIFICATION INSTRUCTIONS 507
ANALYSIS INSTRUCTIONS 509
SUBGRAPHS INSTRUCTIONS 511
INFERENCE INSTUCTIONS 513
GRAPHICAL MODEL INSTUCTIONS 515
PROCESSOR 503

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/032212

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/00(2006.01)i, G06F 21/00(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 17/00; G06F 15/173; G06F 9/44; G06F 21/56; G06F 12/14; H04L 9/32; G06F 21/00; G06F 9/45

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: node, graphical model, time period, snapshot, marginal distribution, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014-0181819 A1 (MICROSOFT CORPORATION) 26 June 2014 See paragraphs [0002]-[0005], [0020]-[0027], and [0043]; claim 12; and figures 3a-3e.	1-15
A	US 2013-0179974 A1 (PRATYUSA KUMAR MANADHATA et al.) 11 July 2013 See paragraphs [0009]-[0015] and [0022]-[0032]; claim 1; and figure 3.	1-15
A	US 9021260 B1 (PALANTIR TECHNOLOGIES INC.) 28 April 2015 See column 1, line 40 - column 2, line 40; and figure 1.	1-15
A	WO 2007-117567 A2 (SMOBILE SYSTEMS INC.) 18 October 2007 See page 4, lines 4-24; page 13, lines 3-27; and figure 2.	1-15
A	US 2007-0016898 A1 (HERBERT G. DERBY et al.) 18 January 2007 See paragraphs [0011]-[0013]; and figure 26.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 April 2016 (20.04.2016)	**22 April 2016 (22.04.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
HAN, JOONG SUB
Telephone No. +82-42-481-3578

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/032212

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0181819 A1	26/06/2014	WO 2014-100785 A1	26/06/2014
US 2013-0179974 A1	11/07/2013	US 9032527 B2	12/05/2015
US 9021260 B1	28/04/2015	None	
WO 2007-117567 A2	18/10/2007	EP 2011099 A2	07/01/2009
		EP 2011099 A4	21/08/2013
		MX 2008012891 A	22/07/2009
		US 2007-240217 A1	11/10/2007
		US 2007-240218 A1	11/10/2007
		US 2007-240219 A1	11/10/2007
		US 2007-240220 A1	11/10/2007
		US 2007-240221 A1	11/10/2007
		US 2007-240222 A1	11/10/2007
		US 2011-179484 A1	21/07/2011
		US 8312545 B2	13/11/2012
		US 8321941 B2	27/11/2012
		US 9009818 B2	14/04/2015
		US 9064115 B2	23/06/2015
		US 9104871 B2	11/08/2015
		WO 2007-117567 A3	28/02/2008
		WO 2007-117574 A2	18/10/2007
		WO 2007-117574 A3	21/08/2008
		WO 2007-117582 A2	18/10/2007
		WO 2007-117582 A3	14/08/2008
		WO 2007-117585 A2	18/10/2007
		WO 2007-117585 A3	15/05/2008
		WO 2007-117635 A2	18/10/2007
		WO 2007-117635 A3	26/06/2008
		WO 2007-117636 A2	18/10/2007
		WO 2007-117636 A3	24/04/2008
US 2007-0016898 A1	18/01/2007	US 7233733 B2	19/06/2007