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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 March 2017 (30.03.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/052959 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2016/048595

(22) International Filing Date:
25 August 2016 (25.08.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/866,190 25 September 2015 (25.09.2015) US

(71) Applicant: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, California 95054 (US).

(72) Inventors: **SUN, Lawrence J.**; 8704 SW Marseilles Drive, Beaverton, Oregon 97007 (US). **TOVINKERE, Vasanth R.**; 15254 NW Mitchell Street, Portland, Oregon 97229 (US).

(74) Agent: **KELLETT, Glen M.**; Barnes & Thornburg LLP, c/o CPA Global, P.O. Box 52050, Minneapolis, Minnesota 55402 (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 applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: TECHNOLOGIES FOR AUTOMATIC PARTITIONING OF LARGE GRAPHS

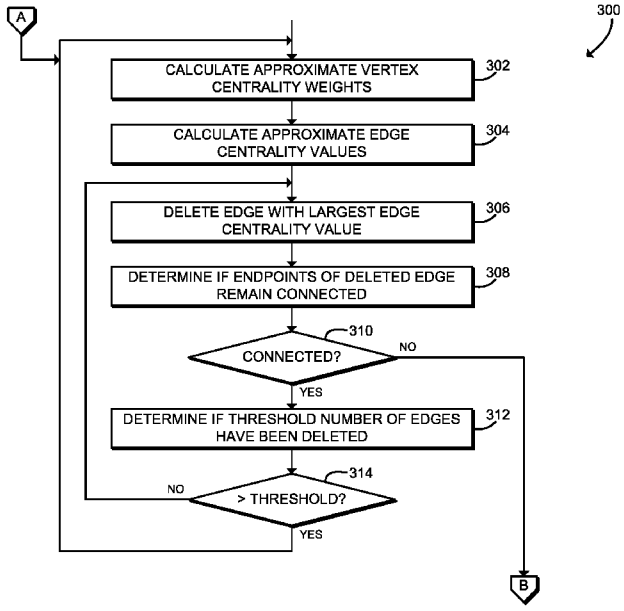

FIG. 3A

(57) Abstract: Technologies for automatic graph partitioning include a computing device that approximates a vertex centrality weight for each vertex of a graph and then approximates, based on the approximate vertex centrality weight, an approximate edge centrality value for each edge of the graph. The computing device may repeatedly delete an edge having the highest edge centrality value and test if the graph has been disconnected. If the graph is disconnected, the computing device calculates a cluster quality metric. If the cluster quality does not decrease, the computing device realizes a new clustering of the graph based on the disconnected partitions. If the cluster quality metric decreases, the computing device reintroduces a deleted edge. The computing device recalculates the approximate vertex centrality weights and edge centrality values after reintroducing a deleted edge, deleting a predefined number of edges, or realizing a new clustering. Other embodiments are described and claimed.

WO 2017/052959 A1

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

TECHNOLOGIES FOR AUTOMATIC PARTITIONING OF LARGE GRAPHS

CROSS-REFERENCE TO RELATED U.S. PATENT APPLICATION

[0001] The present application claims priority to U.S. Utility Patent Application Serial No. 14/866,190, entitled "TECHNOLOGIES FOR AUTOMATIC PARTITIONING OF LARGE GRAPHS," which was filed on September 25, 2015.

BACKGROUND

[0002] In a data-flow programming paradigm, computer programs may be described as data-flow graphs. To perform performance analysis of data-flow programs, the associated data-flow graph may be partitioned into smaller, independent graphs that may be analyzed individually. For example, different partitions of a graph may have different performance characteristics and thus may benefit from different performance solutions.

[0003] Determining an optimal graph partitioning is typically an NP-hard problem, and thus graph partitioning algorithms typically sacrifice either speed or accuracy to obtain satisfactory results. Typical graph partitioning algorithms utilize global properties of the graph. For example, eigenvectors of the Laplacian matrix or the minimum cut of the graph may be used to partition the graph. As another example, graph partitioning may be performed by continually deleting the edge of the graph with the largest edge centrality to disconnect the graph and find the partitions. Typical graph partitioning algorithms such as these do not scale well to large graph sizes.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The concepts described herein are illustrated by way of example and not by way of limitation in the accompanying figures. For simplicity and clarity of illustration, elements illustrated in the figures are not necessarily drawn to scale. Where considered appropriate, reference labels have been repeated among the figures to indicate corresponding or analogous elements.

[0005] FIG. 1 is a simplified block diagram of at least one embodiment of a computing device for automatic graph partitioning;

[0006] FIG. 2 is a simplified block diagram of at least one embodiment of an environment of the computing device of FIG. 1;

[0007] FIGS. 3A and 3B are a simplified flow diagram of at least one embodiment of a method for automatic graph partitioning that may be executed by the computing device of FIGS. 1 and 2;

[0008] FIGS. 4A to 4C are schematic diagrams of illustrative graphs that may be analyzed by the method of FIG. 3;

[0009] FIG. 5 is a simplified flow diagram of at least one embodiment of a method for approximating vertex centrality that may be executed by the computing device of FIGS. 1 and 2; and

[0010] FIG. 6 is a simplified flow diagram of at least one embodiment of a method for approximating edge centrality that may be executed by the computing device of FIGS. 1 and 2.

DETAILED DESCRIPTION OF THE DRAWINGS

[0011] While the concepts of the present disclosure are susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and will be described herein in detail. It should be understood, however, that there is no intent to limit the concepts of the present disclosure to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives consistent with the present disclosure and the appended claims.

[0012] References in the specification to "one embodiment," "an embodiment," "an illustrative embodiment," etc., indicate that the embodiment described may include a particular feature, structure, or characteristic, but every embodiment may or may not necessarily include that particular feature, structure, or characteristic. Moreover, such phrases are not necessarily referring to the same embodiment. Further, when a particular feature, structure, or characteristic is described in connection with an embodiment, it is submitted that it is within the knowledge of one skilled in the art to effect such feature, structure, or characteristic in connection with other embodiments whether or not explicitly described. Additionally, it should be appreciated that items included in a list in the form of "at least one A, B, and C" can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C). Similarly, items listed in the form of "at least one of A, B, or C" can mean (A); (B); (C); (A and B); (A and C); (B and C); or (A, B, and C).

[0013] The disclosed embodiments may be implemented, in some cases, in hardware, firmware, software, or any combination thereof. The disclosed embodiments may also be implemented as instructions carried by or stored on a transitory or non-transitory machine-readable (e.g., computer-readable) storage medium, which may be read and executed by one or more processors. A machine-readable storage medium may be embodied as any storage device, mechanism, or other physical structure for storing or transmitting information in a form

readable by a machine (e.g., a volatile or non-volatile memory, a media disc, or other media device).

[0014] In the drawings, some structural or method features may be shown in specific arrangements and/or orderings. However, it should be appreciated that such specific arrangements and/or orderings may not be required. Rather, in some embodiments, such features may be arranged in a different manner and/or order than shown in the illustrative figures. Additionally, the inclusion of a structural or method feature in a particular figure is not meant to imply that such feature is required in all embodiments and, in some embodiments, may not be included or may be combined with other features.

[0015] Referring now to FIG. 1, an illustrative computing device 100 for automatic partitioning of a computer program graph includes a processor 120, an I/O subsystem 122, a memory 124, and a data storage device 126. In use, as described below, the computing device 100 is configured to calculate approximate vertex centrality weights and approximate edge centrality values for a graph representing a computer program using iterative sampling algorithms. The computing device 100 deletes the edge with the highest edge centrality value in an attempt to disconnect partitions within the graph. The computing device 100 may delete multiple edges before recalculating approximate vertex centrality weights and edge centrality values. When a clustering of the graph is identified, the computing device 100 calculates one or more cluster quality metrics. If a cluster quality metric decreases, the computing device 100 may backtrack by reintroducing a deleted edge to the graph and recalculating approximate vertex centrality weights and edge centrality values. After a threshold number of backtracks, the computing device 100 may ignore the cluster quality metric and proceed without backtracking. Optimal clusterings may be identified as clusterings with the highest cluster quality metric. The automatic graph partitioning performed by the computing device 100 may scale well to large graph sizes, particularly to large sparse graphs representing data-flow oriented computer programs. Thus, the computing device 100 may perform automatic graph partitioning with sufficient performance for many practical uses, including performance analysis of large data-flow graphs, dispatching work in parallel, semantic analysis of large graphs, or other graph analysis tasks.

[0016] The computing device 100 may be embodied as any type of device capable of performing the functions described herein. For example, the computing device 100 may be embodied as, without limitation, a computer, a workstation, a server computer, a laptop computer, a notebook computer, a tablet computer, a smartphone, a mobile computing device, a desktop computer, a distributed computing system, a multiprocessor system, a consumer

electronic device, a smart appliance, and/or any other computing device capable of analyzing software code segments. As shown in FIG. 1, the illustrative computing device 100 includes the processor 120, the I/O subsystem 122, the memory 124, and the data storage device 126. Of course, the computing device 100 may include other or additional components, such as those commonly found in a workstation (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 124, or portions thereof, may be incorporated in the processor 120 in some embodiments.

[0017] The processor 120 may be embodied as any type of processor capable of performing the functions described herein. For example, the processor may be embodied as a single or multi-core processor(s), digital signal processor, microcontroller, or other processor or processing/controlling circuit. Similarly, the memory 124 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In operation, the memory 124 may store various data and software used during operation of the computing device 100 such operating systems, applications, programs, libraries, and drivers. The memory 124 is communicatively coupled to the processor 120 via the I/O subsystem 122, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor 120, the memory 124, and other components of the computing device 100. For example, the I/O subsystem 122 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, firmware devices, communication links (i.e., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.) and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 122 may form a portion of a system-on-a-chip (SoC) and be incorporated, along with the processor 120, the memory 124, and other components of the computing device 100, on a single integrated circuit chip.

[0018] The data storage device 126 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid-state drives, or other data storage devices. The data storage device 126 may store, for example, one or more graphs to be analyzed.

[0019] The computing device 100 may also include a communication subsystem 128, which may be embodied as any communication circuit, device, or collection thereof, capable of enabling communications between the computing device 100 and other remote devices over a computer network (not shown). The communication subsystem 128 may be configured to use any one or more communication technology (e.g., wired or wireless communications) and

associated protocols (e.g., Ethernet, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0020] Additionally, the computing device 100 may include a display 130 that may be embodied as any type of display capable of displaying digital information such as a liquid crystal display (LCD), a light emitting diode (LED), a plasma display, a cathode ray tube (CRT), or other type of display device. In some embodiments, the computing device 100 may also include one or more peripheral devices 132. The peripheral devices 132 may include any number of additional input/output devices, interface devices, and/or other peripheral devices.

[0021] Referring now to FIG. 2, in the illustrative embodiment, the computing device 100 establishes an environment 200 during operation. The illustrative embodiment 200 includes a centrality module 202, a deletion module 208, and a cluster module 210. The various modules of the environment 200 may be embodied as hardware, firmware, software, or a combination thereof. As such, in some embodiments, one or more of the modules of the environment 200 may be embodied as circuitry or collection of electrical devices (e.g., centrality circuitry 202, deletion circuitry 208, and/or cluster circuitry 210). It should be appreciated that, in such embodiments, one or more of the centrality circuitry 202, the deletion circuitry 208, and/or the cluster circuitry 210 may form a portion of one or more of the processor 120, the I/O subsystem 122, the memory 124, the data storage 126, the communication subsystem 128, and/or other components of the computing device 100. Additionally, in some embodiments, one or more of the illustrative modules may form a portion of another module and/or one or more of the illustrative modules may be independent of one another.

[0022] As shown, the environment 200 includes graph data 214, which may represent a data flow graph 214 or other graph representing a computer program that is to be analyzed. The illustrative graph data 214 includes vertices 216 and edges 218. The centrality module 202 is configured to calculate approximate vertex centrality weights 204 for each vertex 216 of the graph 214. The centrality module 202 is further configured to calculate approximate edge centrality values 206 for each edge 218 of the graph 214 based on the vertex centrality weights 204. As described further below, the centrality module 202 is further configured recalculate the approximate vertex centrality weights 204 and the approximate edge centrality values 206 if a threshold number of edges 218 have been deleted from the graph 214, if a deleted edge 218 is reintroduced into the graph 214, or if a new clustering of the graph 214 is realized.

[0023] The deletion module 208 is configured to delete an edge 218 of the graph 214 having the largest approximate edge centrality value 206 of the graph 214. The deletion

module 208 is further configured to determine whether the former endpoints 216 of the deleted edge 218 are connected in the graph 214 subsequent to deleting the edge 218. The deletion module 208 is further configured to continue deleting edges 218 of the graph 214 having the highest edge centrality value 206 until the endpoints 216 of the deleted edge 218 are no longer connected or until a threshold number of edges 218 have been deleted.

[0024] The cluster module 210 is configured to compute one or more cluster quality metrics 212 for the graph 214 after it is determined the former endpoints 216 of a deleted edge 218 are not connected in the graph 214. As described below, the cluster quality metrics data 212 may include a modularity metric and/or a modified cluster path length. The cluster module 210 is further configured to determine whether a cluster quality metric 212 has decreased in response to determining that the former endpoints 216 of a deleted edge 218 are not connected in the graph 214 214. If the cluster quality metric 212 has not decreased, then a new clustering of the graph 214 has been realized. The cluster module 210 is configured to record the current clustering and the associated cluster quality metric 212. When a sufficient number of clusterings have been recorded, the cluster module 210 is configured to identify an optimal clustering of the graph 214 for each of the associated cluster quality metrics 212. If it is determined that the cluster quality metric 212 has decreased, the deletion module 208 is configured to reintroduce the deleted edge 218 into the graph 214 and increment a backtrack counter. If a predetermined maximum backtrack threshold is exceeded, the cluster module 210 may be configured to record the current clustering and the associated cluster quality metric 212 even if the cluster quality metric 212 has decreased.

[0025] Referring now to FIGS. 3A and 3B, in use, the computing device 100 may execute a method 300 for automatic partitioning of a computer program graph. The method 300 begins in block 302, in which the computing device 100 calculates approximate vertex centrality weights 204 for each vertex 216 in a graph 214. As described above, the graph 214 may be embodied as, for example, a data-flow graph that represents a computer program. Vertex centrality measures the number of shortest paths between all vertices 216 that include a particular vertex 216. The vertices 216 that are included in more shortest paths are typically more central to the graph 214 and thus may indicate boundaries between partitions in the graph 214. To approximate the vertex centrality weights 204, the computing device 100 may perform an iterative sampling algorithm as described below in connection with FIG. 5.

[0026] In block 304, the computing device 100 calculates approximate edge centrality values 206 for each edge 218 in the graph 214. The computing device 100 calculates the approximate edge centrality based on the vertex centrality weights 204 determined as described

above in connection with block 302. Edge centrality measures the number of shortest paths between all vertices 216 that include a particular edge 218. The edges 218 that are included in more shortest paths are typically more central to the graph 214 and thus may indicate boundaries between partitions in the graph 214. To approximate the edge centrality values 206, the computing device 100 may perform an iterative sampling algorithm as described below in connection with FIG. 6.

[0027] In block 306, the computing device 100 deletes the edge 218 with the highest edge centrality value 206 from the graph 214. The computing device 100 may record the deleted edge 218 so that the deleted edge 218 may be reintroduced into the graph 214 (i.e., "undeleted") as described further below. In block 308, the computing device 100 determines if the endpoints 216 of the deleted edge 218 remain connected in the graph 214. The computing device 100 may, for example, determine if any path exists in the graph 214 between the vertices 216 that were the endpoints of the deleted edge 218. In some embodiments, the computing device 100 may start a breadth-first search of the graph 214 from one of the endpoints 216 and determine whether the other endpoint 216 is reachable. In block 310, the computing device 100 checks whether the endpoints 216 of the deleted edge 218 remain connected in the graph 214. If not, the method 300 branches ahead to block 316, shown in FIG. 3B and described further below. If the endpoints 216 of the deleted edge 218 remain connected, the method 300 advances to block 312.

[0028] In block 312, the computing device 100 determines if a threshold number of edges 218 have been deleted from the graph 214. For example, the computing device 100 may check a counter that is incremented after every deletion of an edge 218. In the illustrative embodiment, the threshold number of edges is five edges, which provides the greatest speed increase to the algorithm without causing excessive backtracking due to deleting too many edges 218 prior to recalculating the edge centrality values. In block 314, the computing device 100 checks whether the threshold number of deletions has been exceeded. If not, the method 300 loops back to block 306 to continue deleting the edge 218 with the largest edge centrality value 206. If the threshold has been exceeded, the method 300 loops back to block 302 to recalculate the approximate vertex centrality weights 204 and the approximate edge centrality values 206. The computing device 100 may also reset an edge deletion counter or otherwise reset the number of edges 218 that have been deleted.

[0029] Referring now to FIG. 4A, schematic diagram 400 illustrates one potential embodiment of a graph 214. The illustrative graph 214 includes eight vertices 216, labeled v_1

to v_8. In the illustrative diagram 400, the edges 218 are labeled e_{ij}, where i and j identify the endpoints of the labeled edge (i.e., the vertices 216 that are connected by the edge 218). For example, the edge e_{12} connects vertices v_1 and v_2, the edge e_{13} connects vertices v_1 and v_3, and so on. In the illustrative embodiment, the computing device 100 may determine that the edge e_{34} has the highest edge centrality value 206. Referring now to FIG. 4B, schematic diagram 402 illustrates the graph 214 after the edge e_{34} has been deleted. The endpoints of the deleted edge e_{34} are the vertices v_3 and v_4. As shown, a path 404 exists between the vertices v_3 and v_4. Thus, after deletion, the former endpoints of the edge e_{34} remain connected in the graph 214. After deletion of the edge e_{34}, the computing device 100 may determine that the edge e_{36} has the highest edge centrality value 206 of the edges 218 remaining in the graph 214. Referring now to FIG. 4C, schematic diagram 406 illustrates the graph 214 after the edge e_{36} has been deleted. The endpoints of the deleted edge e_{36} are the vertices v_3 and v_6. As shown in FIG. 4C, no path exists between the vertices v_3 and v_6. Thus, after deletion of the edge e_{36}, the graph 214 has been disconnected into two partitions 408, 410.

[0030] Referring back FIGS. 3A and 3B, as described above in connection with block 310, if the endpoints 216 of the deleted edge 218 are not connected, the method 300 branches to block 316, shown in FIG. 3B. Once the endpoints 216 of the deleted edge 218 are no longer connected in the graph 214, a new clustering of the graph 214 has been identified. The two disjoint portions of the graph 214 connected to each endpoint 216 of the deleted edge 218 are the new clusters, which may replace a previously-identified cluster that included the two endpoints 216.

[0031] In block 316, the computing device 100 computes one or more cluster quality metrics 212 for the graph 214. The cluster quality metric 212 may be embodied as any measure indicating a characteristic of the current clustering within the graph 214, such as connectedness, compactness, or other characteristics. In some embodiments, in block 318, the computing device 100 may compute a modularity metric Q for the graph 214. The computing device 100 may compute the modularity metric Q using Equation 1, as shown below. To calculate Q, the computing device 100 constructs a $k \times k$ matrix $\mathbf{e}$, where k is the number of clusters or partitions found in the graph 214. Each element e_{ij} of $\mathbf{e}$ is the fraction of all edges 218 in the graph 214 that link vertices 216 in the cluster i to vertices 216 in the cluster j. The values of the matrix $\mathbf{e}$ are determined based on the original graph 214, prior to any deletions of edges 218.

The row sums a_i are defined as $a_i = \sum_j e_{ij}$, and represent the fraction of all edges 218 that connect to vertices 216 in the cluster *i*. As shown in Equation 1, the modularity measure Q equals the trace of the matrix $\mathbf{e}$ minus the sum of the elements of the matrix $\mathbf{e}^2$. Values of Q close to zero indicate that the number of within-cluster edges is no better than would be expected with random connections between vertices, and thus may indicate poor clustering. Values of Q close to one, which is the maximum value, may indicate good clustering.

$$Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\mathbf{e} - \left\|\mathbf{e}^2\right\| \tag{1}$$

[0032] In some embodiments, in block 320 the computing device 100 may calculate a modified cluster path length metric for the graph 214. The modified cluster path metric calculated by the computing device 100 maximizes at reasonable cluster numbers and sizes for large graphs and peaks for similar clusterings as compared to the modularity metric Q. Similar to the modularity metric Q, higher values of the modified cluster length indicate better clustering. As shown in Equation 2, the modified cluster path length M is equal to a plus component M^+ minus four times a minus component M^-.

$$M = M^+ - 4M^- \tag{2}$$

The plus component M^+ is calculated as shown in Equation 3. The term n_i of Equation 3 represents the number of vertices 216 in a cluster *i*, and the term *n* represents the number of vertices 216 in the graph 214. Thus, the plus component M^+ equals the sum of average distance between vertices 216 in the graph 214 over the average distance between vertices 216 in each cluster, weighted by the relative number of vertices 216 in each cluster.

$$M^+ = \frac{1}{n}\sum_{i=1}^{k} \frac{n_i \cdot \textit{Average Distance Between Vertices in Graph}}{\textit{Average Distance Between Vertices in Cluster } i} \tag{3}$$

The minus component M^- is calculated as shown in Equation 4. As shown, the minus component M^- includes edge density, which may be calculated as shown in Equation 5. The edge density represents the ratio of the number of edges 218 in the graph 214 in relation to the maximum potential number of edges that could be included in the graph 214. Including the edge density in the minus component M^- may prevent over-clustering for sparse graphs 214.

$$M^- = \frac{1}{k(k-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{k} \frac{\textit{Edges Between Clusters } i,j}{n_i \cdot n_j \cdot \textit{Edge Density}} \tag{4}$$

$$\textit{Edge Density} = \frac{\textit{Number of Edges}}{n(n-1)} \tag{5}$$

[0033] After calculating the cluster quality metrics 212, in block 322 the computing device 100 determines whether a maximum number of backtracks has been exceeded. As described further below, in certain circumstances the computing device 100 may backtrack by reintroducing a deleted edge 218 to the graph 214. Each time the computing device 100 reintroduces a deleted edge 218, the computing device 100 may increment a backtrack counter. Thus, in block 322 the computing device 100 may compare the backtrack counter to a predetermined threshold number of backtracks. In the illustrative embodiment, the maximum number of backtracks is 10, which provides a satisfactory balance of execution speed and cluster quality. A larger maximum number of backtracks may improve cluster quality while reducing execution speed. If the maximum number of backtracks is exceeded, the method 300 branches ahead to block 332, described below. If the maximum number of backtracks is not exceeded, the method 300 advances to block 324.

[0034] In block 324, the computing device 100 determines whether any of the cluster quality metrics 212 has decreased significantly as a result of deleting the edges 218. The cluster quality metrics 212 may be initialized to a minimum value, such as zero, and the computing device 100 may maintain previous values of the cluster quality metrics 212 for previous clusterings of the graph 214. A decreasing cluster quality metric 212 indicates that the current clustering of the graph 214 (i.e., splitting a previously identified cluster into two newly identified clusters) has caused a drop in cluster quality rather than an increase in cluster quality. The computing device 100 may determine whether a drop in the cluster quality metric 212 has exceeded a predefined threshold. If a cluster quality metric 212 has not decreased significantly, the method 300 branches ahead to block 332, described below. If a cluster quality metric 212 has decreased significantly, the method 300 advances to block 328.

[0035] In block 328, the computing device 100 reintroduces the most recently deleted edge 218 to the graph 214. For example, referring to FIG. 4C, if the computing device 100 determines that a cluster quality metric 212 associated with the clusters 408, 410 has dropped, the computing device 100 may reintroduce the edge e_{36}, resulting in the graph 214 as shown in FIG. 4B. Referring back to FIG. 3B, in block 330, after reintroducing the deleted edge 218 the computing device 100 increments the backtrack counter. As described above in connection with block 322, if the maximum number of backtracks is exceeded, the method 300 skips ahead to block 332 without determining whether the quality metric has decreased. Thus, by attempting a maximum number of backtracks and then skipping ahead, the method 300 may continue to make progress even if the cluster quality metric 212 continues to decrease after multiple backtracks. After incrementing the backtrack counter, the method 300 loops back to

block 302 shown in FIG. 3A to recalculate the approximate vertex centrality weights 204 and the edge centrality values 206.

[0036] Referring back to blocks 322, 326, if the cluster quality metric 212 has not decreased—or if the maximum number of backtracks has been exceeded—the method 300 branches ahead to block 332. In block 332, the computing device 100 resets the backtrack counter, indicating that a new clustering has been realized. In block 334, the computing device 100 records the current clustering of the graph 214 as well as the associated cluster quality metrics 212. The computing device 100 may record, for example, the vertices 216 included in each identified cluster within the graph 214. In block 336, the computing device 100 determines whether a sufficient number of clusterings have been recorded. For example, the computing device 100 may determine whether the cluster quality metrics 212 have decreased for several iterations, leveled off, or otherwise stabilized. As another example, the computing device 100 may continue finding clusters and recording clusterings until a predefined number of clusters has been identified, or until the identified clusters have a predefined size. If sufficient clusterings have not been recorded, the method 300 loops back to block 302 shown in FIG. 3A to recalculate the approximate vertex centrality weights 204 and the edge centrality values 206. If sufficient clusterings have been recorded, the method 300 advances to block 338.

[0037] In block 338, the computing device 100 identifies optimal clusterings of the graph 214 as the clusterings having the highest associated cluster quality metrics 212. As described above, each clustering may identify the vertices 216 of the graph 214 that are included in each cluster. The computing device 100 may identify an optimal clustering for each cluster quality metric 212. For example, in the illustrative embodiment, the computing device 100 may identify two optimal clusterings: one clustering for the modularity metric Q and another clustering for the modified cluster path length M. For many graphs 214, the optimal clusterings for each of the cluster quality metrics 212 may be the same.

[0038] After identifying optimal clustering, the method 300 is completed. The computing device 100 may use the clusterings identified using the method 300 to perform additional analysis of the graph 214. For a data-flow graph 214, breaking the graph 214 into related subsets may enable further understanding, analyzing, and solving of performance problems. For example, identifying partitions of the graph 214 may allow performance analysis of large data-flow and dependence graphs executed by parallel runtimes. As another example, identifying partitions of the graph 214 may allow large graphs to be partitioned and then distributed across multiple machines or other computing devices. Identifying partitions of the

graph 214 may also allow semantic analysis of large graphs and partitioning large graphs into groups with similar characteristics.

[0039] Referring now to FIG. 5, in use, the computing device 100 may execute a method 500 for approximating vertex centrality. As described above, the method 500 may be executed in connection with block 302 of FIG. 3. The method 500 begins in block 502, in which the computing device 100 initializes to zero the vertex centrality weight 204 for each vertex 216 in the graph 214. In block 504, the computing device 100 determines a total number of iterations r to process. The number of iterations r is equal to $n^{\frac{2}{3}}$, that is, the number of vertices 216 in the graph 214 to the two-thirds power. This number of iterations r is selected to provide results with a satisfactorily low expected error compared to the exact betweenness centrality values for the vertices 216. For example, referring to FIG. 4A, the graph 214 includes eight vertices 216. Thus, the number of iterations r for the illustrative graph 214 may be four.

[0040] In block 506, the computing device 100 samples a pair (u, v) of distinct vertices 216 within the graph 214. The computing device 100 selects the vertices 216 uniformly at random. In block 508, the computing device 100 computes the set S_{uv} of all shortest paths between u and v in the graph 214. For example, referring again to FIG. 4A, the computing device 100 may select the vertices v_2 and v_5 at random. In that example, the set S_{uv} of all shortest paths between v_2 and v_5 may include a path p_1 including the vertices (v_2, v_3, v_4, v_5) and a path p_2 including the vertices (v_2, v_3, v_6, v_5).

[0041] In block 510, the computing device 100 selects a path p from the set S_{uv} uniformly at random. In block 512, the computing device 100 increments the vertex centrality weight 204 of each interior vertex 216 of the selected path p by $\frac{1}{r}$. For example, referring again to the FIG. 4A, for the set S_{uv} of all shortest paths between v_2 and v_5, the computing device 100 may randomly select the path p including the vertices (v_2, v_3, v_4, v_5). In that example, the interior vertices 216 of the path p are the vertices (v_3, v_4). Thus, in the illustrative example, the computing device 100 may increment the vertex centrality weight 204 of each interior vertex (v_3, v_4) by $\frac{1}{r}$, which equals 0.25 in the illustrative embodiment.

[0042] In block 514, the computing device 100 determines whether r iterations have been processed. If not, the method 500 loops back to block 506 to sample another pair of distinct vertices 216. If r iterations have been processed, the method 500 is completed, and approximate vertex centrality weights 204 have been calculated for the graph 214.

[0043] Referring now to FIG. 6, in use, the computing device 100 may execute a method 600 for approximating edge centrality. As described above, the method 600 may be executed in connection with block 304 of FIG. 3. The method 600 begins in block 602, in which the computing device 100 initializes to zero the edge centrality value 206 for each edge 218 in the graph 214. In block 604, the computing device 100 determines a total number of iterations r to process. The number of iterations r is equal to $n^{\frac{2}{3}}$, that is, the number of vertices 216 in the graph 214 to the two-thirds power. For example, referring to FIG. 4A, the graph 214 includes eight vertices 216. Thus, the number of iterations r for the illustrative graph 214 may be four.

[0044] In block 606, the computing device 100 samples a pair $(u, v,)$ of distinct vertices 216 within the graph 214. The computing device 100 selects the vertices u, v with a probability that is weighted according to the vertex centrality weights 204 associated with the vertices 216. In other words, vertices 216 with a higher associated vertex centrality weight 204 are more likely to be sampled. Thus, unlike for the approximation of the vertex centrality weights 204 described above in connection with FIG. 5, the pair of vertices u, v are not selected uniformly at random. In block 608, the computing device 100 computes the set S_{uv} of all shortest paths between u and v in the graph 214. For example, referring again to FIG. 4A, the computing device 100 may select the vertices v_2 and v_5. In that example, the set S_{uv} of all shortest paths between v_2 and v_5 may include a path p_1 including the vertices (v_2, v_3, v_4, v_5) and a path p_2 including the vertices (v_2, v_3, v_6, v_5).

[0045] In block 610, the computing device 100 selects a path p from the set S_{uv} uniformly at random. In block 612, the computing device 100 increments the edge centrality value 206 of each edge 218 in the selected path p by $\frac{1}{r}$. For example, referring again to the FIG. 4A, for the set S_{uv} of all shortest paths between v_2 and v_5, the computing device 100 may randomly select the path p including the vertices (v_2, v_3, v_4, v_5). In that example, the edges of the selected path p are the edges (e_{23}, e_{34}, e_{45}). Thus, in the illustrative example, the computing device 100 may increment the edge centrality value 206 of each edge (e_{23}, e_{34}, e_{45}) by $\frac{1}{r}$, which equals 0.25 in the illustrative embodiment.

[0046] In block 614, the computing device 100 determines whether r iterations have been processed. If not, the method 600 loops back to block 606 to sample another pair of distinct vertices 216. If r iterations have been processed, the method 600 is completed, and approximate edge centrality values 206 have been calculated for the graph 214.

[0047] It should be appreciated that, in some embodiments, any one or more of the methods 300, 500, and/or 600 may be embodied as various instructions stored on a computer-readable media, which may be executed by the processor 120, a peripheral device 132, and/or other components of the computing device 100 to cause the computing device 100 to perform the corresponding method 300, 500, and/or 600. The computer-readable media may be embodied as any type of media capable of being read by the computing device 100 including, but not limited to, the memory 124, the data storage 126, a local memory of the processor 120, other memory or data storage devices of the computing device 100, portable media readable by a peripheral device 132 of the computing device 100, and/or other media.

EXAMPLES

[0048] Illustrative examples of the technologies disclosed herein are provided below. An embodiment of the technologies may include any one or more, and any combination of, the examples described below.

[0049] Example 1 includes a computing device for automatic graph partitioning, the computing device comprising centrality circuitry to (i) calculate an approximate vertex centrality weight for each vertex of a graph and (ii) calculate, based on the approximate vertex centrality weight for each vertex of the graph, an approximate edge centrality value for each edge of the graph; deletion circuitry to (i) delete a first edge of the graph, wherein the first edge connects a first vertex and a second vertex, and wherein the first edge has a largest approximate edge centrality value of the edges of the graph and (ii) determine whether the first vertex and the second vertex are connected in the graph subsequent to deletion of the first edge; and cluster circuitry to compute a cluster quality metric for the graph in response to a determination that the first vertex and the second vertex are not connected in the graph subsequent to deletion of the first edge.

[0050] Example 2 includes the subject matter of Example 1, and wherein the deletion circuitry is further to (i) determine whether a threshold number of edges have been deleted in response to a determination that the first vertex and the second vertex remain connected in the graph and (ii) delete a second edge of the graph in response to a determination that the threshold number of edges have not been deleted, wherein the second edge has a largest approximate edge centrality value of the edges remaining in the graph; and the centrality circuitry is further to recalculate an approximate vertex centrality weight for each vertex of the graph in response to a determination that the threshold number of edges have been deleted.

[0051] Example 3 includes the subject matter of any of Examples 1 and 2, and wherein the threshold number of edges comprises five edges.

[0052] Example 4 includes the subject matter of any of Examples 1-3, and wherein the cluster circuitry is further to (i) determine whether the cluster quality metric has decreased in response to deletion of the first edge of the graph and (ii) record a current clustering of the graph and the cluster quality metric in response to a determination the cluster quality metric has not decreased.

[0053] Example 5 includes the subject matter of any of Examples 1-4, and wherein the cluster circuitry is further to (i) determine whether sufficient clusterings have been recorded in response to recordation of the current clustering of the graph and (ii) identify an optimal clustering of the graph based on the cluster quality metric in response to a determination that sufficient clusterings have been recorded.

[0054] Example 6 includes the subject matter of any of Examples 1-5, and wherein the deletion circuitry is further to reintroduce the first edge into the graph in response to a determination that the cluster quality metric has decreased; and the centrality circuitry is further to recalculate an approximate vertex centrality weight for each vertex of the graph in response to reintroduction of the first edge into the graph.

[0055] Example 7 includes the subject matter of any of Examples 1-6, and wherein the deletion circuitry is further to (i) determine whether a backtrack counter exceeds a predetermined backtrack threshold in response to the determination that the cluster quality metric has decreased and (ii) increment the backtrack counter in response to the reintroduction of the first edge into the graph; and to reintroduce the first edge into the graph further comprises to reintroduce the first edge into the graph in response to a determination that the backtrack counter does not exceed the predetermined backtrack threshold.

[0056] Example 8 includes the subject matter of any of Examples 1-7, and wherein to record the cluster quality metric for the current clustering of the graph comprises to record the cluster quality metric for the current clustering of the graph in response to the determination that the cluster quality metric has not decreased or a determination that the backtrack counter exceeds the predetermined backtrack threshold.

[0057] Example 9 includes the subject matter of any of Examples 1-8, and wherein the predetermined backtrack threshold comprises ten backtracks.

[0058] Example 10 includes the subject matter of any of Examples 1-9, and wherein the cluster quality metric comprises a modularity metric.

[0059] Example 11 includes the subject matter of any of Examples 1-10, and wherein the cluster quality metric comprises a modified cluster path length.

[0060] Example 12 includes the subject matter of any of Examples 1-11, and wherein to compute the modified cluster path length comprises to weight a ratio of average distance between vertices in the graph and average distance between vertices in a cluster by a number of vertices in the cluster.

[0061] Example 13 includes the subject matter of any of Examples 1-12, and wherein to compute the modified cluster path length comprises to compute the modified cluster path length as a function of an edge density of the graph.

[0062] Example 14 includes the subject matter of any of Examples 1-13, and wherein to calculate the approximate vertex centrality weight for each vertex of the graph comprises to select a pair of distinct vertices from the graph uniformly at random; compute a set of all shortest paths between the pair of distinct vertices; select a path from the set of all shortest paths uniformly at random; and increment the approximate vertex centrality weight of each interior vertex of the path.

[0063] Example 15 includes the subject matter of any of Examples 1-14, and wherein to calculate, based on the approximate vertex centrality weight for each vertex of the graph, the approximate edge centrality value for each edge of the graph comprises to select a pair of distinct vertices from the graph with a probability of each vertex equal to the approximate vertex centrality weight of the corresponding vertex; compute a set of all shortest paths between the pair of distinct vertices; select a path from the set of all shortest paths uniformly at random; and increment the approximate edge centrality value of each edge of the path.

[0064] Example 16 includes a method for automatic graph partitioning, the method comprising calculating, by a computing device, an approximate vertex centrality weight for each vertex of a graph; calculating, by the computing device and based on the approximate vertex centrality weight for each vertex of the graph, an approximate edge centrality value for each edge of the graph; deleting, by the computing device, a first edge of the graph, wherein the first edge connects a first vertex and a second vertex, and wherein the first edge has a largest approximate edge centrality value of the edges of the graph; determining, by the computing device, whether the first vertex and the second vertex are connected in the graph subsequent to deleting the first edge; and computing, by the computing device, a cluster quality metric for the graph in response to determining that the first vertex and the second vertex are not connected in the graph subsequent to deleting the first edge.

[0065] Example 17 includes the subject matter of Example 16, and further including determining, by the computing device, whether a threshold number of edges have been deleted in response to determining that the first vertex and the second vertex remain connected in the graph; deleting, by the computing device, a second edge of the graph in response to determining that the threshold number of edges have not been deleted, wherein the second edge has a largest approximate edge centrality value of the edges remaining in the graph; and recalculating, by the computing device, an approximate vertex centrality weight for each vertex of the graph in response to determining that the threshold number of edges have been deleted.

[0066] Example 18 includes the subject matter of any of Examples 16 and 17, and wherein the threshold number of edges comprises five edges.

[0067] Example 19 includes the subject matter of any of Examples 16-18, and further including determining, by the computing device, whether the cluster quality metric has decreased in response to deleting the first edge of the graph; and recording, by the computing device, a current clustering of the graph and the cluster quality metric in response to determining the cluster quality metric has not decreased.

[0068] Example 20 includes the subject matter of any of Examples 16-19, and further including determining, by the computing device, whether sufficient clusterings have been recorded in response to recording the current clustering of the graph; and identifying, by the computing device, an optimal clustering of the graph based on the cluster quality metric in response to determining that sufficient clusterings have been recorded.

[0069] Example 21 includes the subject matter of any of Examples 16-20, and further including reintroducing, by the computing device, the first edge into the graph in response to determining that the cluster quality metric has decreased; and recalculating, by the computing device, an approximate vertex centrality weight for each vertex of the graph in response to reintroducing the first edge into the graph.

[0070] Example 22 includes the subject matter of any of Examples 16-21, and further including determining, by the computing device, whether a backtrack counter exceeds a predetermined backtrack threshold in response to determining that the cluster quality metric has decreased; and incrementing, by the computing device, the backtrack counter in response to reintroducing the first edge into the graph; wherein reintroducing the first edge into the graph further comprises reintroducing the first edge into the graph in response to determining that the backtrack counter does not exceed the predetermined backtrack threshold.

[0071] Example 23 includes the subject matter of any of Examples 16-22, and wherein recording the cluster quality metric for the current clustering of the graph comprises recording

the cluster quality metric for the current clustering of the graph in response to determining the cluster quality metric has not decreased or determining that the backtrack counter exceeds the predetermined backtrack threshold.

[0072] Example 24 includes the subject matter of any of Examples 16-23, and wherein the predetermined backtrack threshold comprises ten backtracks.

[0073] Example 25 includes the subject matter of any of Examples 16-24, and wherein computing the cluster quality metric comprises computing a modularity metric.

[0074] Example 26 includes the subject matter of any of Examples 16-25, and wherein computing the cluster quality metric comprises computing a modified cluster path length.

[0075] Example 27 includes the subject matter of any of Examples 16-26, and wherein computing the modified cluster path length comprises weighting a ratio of average distance between vertices in the graph and average distance between vertices in a cluster by a number of vertices in the cluster.

[0076] Example 28 includes the subject matter of any of Examples 16-27, and wherein computing the modified cluster path length comprises computing the modified cluster path length as a function of an edge density of the graph.

[0077] Example 29 includes the subject matter of any of Examples 16-28, and wherein calculating the approximate vertex centrality weight for each vertex of the graph comprises selecting a pair of distinct vertices from the graph uniformly at random; computing a set of all shortest paths between the pair of distinct vertices; selecting a path from the set of all shortest paths uniformly at random; and incrementing the approximate vertex centrality weight of each interior vertex of the path.

[0078] Example 30 includes the subject matter of any of Examples 16-29, and wherein calculating, based on the approximate vertex centrality weight for each vertex of the graph, the approximate edge centrality value for each edge of the graph comprises selecting a pair of distinct vertices from the graph with a probability of each vertex equal to the approximate vertex centrality weight of the corresponding vertex; computing a set of all shortest paths between the pair of distinct vertices; selecting a path from the set of all shortest paths uniformly at random; and incrementing the approximate edge centrality value of each edge of the path.

[0079] Example 31 includes a computing device comprising a processor; and a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of Examples 16-30.

[0080] Example 32 includes one or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of Examples 16-30.

[0081] Example 33 includes a computing device comprising means for performing the method of any of Examples 16-30.

[0082] Example 34 includes a computing device for automatic graph partitioning, the computing device comprising means for calculating an approximate vertex centrality weight for each vertex of a graph; means for calculating, based on the approximate vertex centrality weight for each vertex of the graph, an approximate edge centrality value for each edge of the graph; means for deleting a first edge of the graph, wherein the first edge connects a first vertex and a second vertex, and wherein the first edge has a largest approximate edge centrality value of the edges of the graph; means for determining whether the first vertex and the second vertex are connected in the graph subsequent to deleting the first edge; and means for computing a cluster quality metric for the graph in response to determining that the first vertex and the second vertex are not connected in the graph subsequent to deleting the first edge.

[0083] Example 35 includes the subject matter of Example 34, and further including means for determining whether a threshold number of edges have been deleted in response to determining that the first vertex and the second vertex remain connected in the graph; means for deleting a second edge of the graph in response to determining that the threshold number of edges have not been deleted, wherein the second edge has a largest approximate edge centrality value of the edges remaining in the graph; and means for recalculating an approximate vertex centrality weight for each vertex of the graph in response to determining that the threshold number of edges have been deleted.

[0084] Example 36 includes the subject matter of any of Examples 34 and 35, and wherein the threshold number of edges comprises five edges.

[0085] Example 37 includes the subject matter of any of Examples 34-36, and further including means for determining whether the cluster quality metric has decreased in response to deleting the first edge of the graph; and means for recording a current clustering of the graph and the cluster quality metric in response to determining the cluster quality metric has not decreased.

[0086] Example 38 includes the subject matter of any of Examples 34-37, and further including means for determining whether sufficient clusterings have been recorded in response to recording the current clustering of the graph; and means for identifying an optimal clustering

of the graph based on the cluster quality metric in response to determining that sufficient clusterings have been recorded.

[0087] Example 39 includes the subject matter of any of Examples 34-38, and further including means for reintroducing the first edge into the graph in response to determining that the cluster quality metric has decreased; and means for recalculating an approximate vertex centrality weight for each vertex of the graph in response to reintroducing the first edge into the graph.

[0088] Example 40 includes the subject matter of any of Examples 34-39, and further including means for determining whether a backtrack counter exceeds a predetermined backtrack threshold in response to determining that the cluster quality metric has decreased; and means for incrementing the backtrack counter in response to reintroducing the first edge into the graph; wherein the means for reintroducing the first edge into the graph further comprises means for reintroducing the first edge into the graph in response to determining that the backtrack counter does not exceed the predetermined backtrack threshold.

[0089] Example 41 includes the subject matter of any of Examples 34-40, and wherein the means for recording the cluster quality metric for the current clustering of the graph comprises means for recording the cluster quality metric for the current clustering of the graph in response to determining the cluster quality metric has not decreased or determining that the backtrack counter exceeds the predetermined backtrack threshold.

[0090] Example 42 includes the subject matter of any of Examples 34-41, and wherein the predetermined backtrack threshold comprises ten backtracks.

[0091] Example 43 includes the subject matter of any of Examples 34-42, and wherein the means for computing the cluster quality metric comprises means for computing a modularity metric.

[0092] Example 44 includes the subject matter of any of Examples 34-43, and wherein the means for computing the cluster quality metric comprises means for computing a modified cluster path length.

[0093] Example 45 includes the subject matter of any of Examples 34-44, and wherein the means for computing the modified cluster path length comprises means for weighting a ratio of average distance between vertices in the graph and average distance between vertices in a cluster by a number of vertices in the cluster.

[0094] Example 46 includes the subject matter of any of Examples 34-45, and wherein the means for computing the modified cluster path length comprises means for computing the modified cluster path length as a function of an edge density of the graph.

[0095] Example 47 includes the subject matter of any of Examples 34-46, and wherein the means for calculating the approximate vertex centrality weight for each vertex of the graph comprises means for selecting a pair of distinct vertices from the graph uniformly at random; means for computing a set of all shortest paths between the pair of distinct vertices; means for selecting a path from the set of all shortest paths uniformly at random; and means for incrementing the approximate vertex centrality weight of each interior vertex of the path.

Example 48 includes the subject matter of any of Examples 34-47, and wherein the means for calculating, based on the approximate vertex centrality weight for each vertex of the graph, the approximate edge centrality value for each edge of the graph comprises means for selecting a pair of distinct vertices from the graph with a probability of each vertex equal to the approximate vertex centrality weight of the corresponding vertex; means for computing a set of all shortest paths between the pair of distinct vertices; means for selecting a path from the set of all shortest paths uniformly at random; and means for incrementing the approximate edge centrality value of each edge of the path.

WHAT IS CLAIMED IS:

1. A computing device for automatic graph partitioning, the computing device comprising:

centrality circuitry to (i) calculate an approximate vertex centrality weight for each vertex of a graph and (ii) calculate, based on the approximate vertex centrality weight for each vertex of the graph, an approximate edge centrality value for each edge of the graph;

deletion circuitry to (i) delete a first edge of the graph, wherein the first edge connects a first vertex and a second vertex, and wherein the first edge has a largest approximate edge centrality value of the edges of the graph and (ii) determine whether the first vertex and the second vertex are connected in the graph subsequent to deletion of the first edge; and

cluster circuitry to compute a cluster quality metric for the graph in response to a determination that the first vertex and the second vertex are not connected in the graph subsequent to deletion of the first edge.

2. The computing device of claim 1, wherein:

the deletion circuitry is further to (i) determine whether a threshold number of edges have been deleted in response to a determination that the first vertex and the second vertex remain connected in the graph and (ii) delete a second edge of the graph in response to a determination that the threshold number of edges have not been deleted, wherein the second edge has a largest approximate edge centrality value of the edges remaining in the graph; and

the centrality circuitry is further to recalculate an approximate vertex centrality weight for each vertex of the graph in response to a determination that the threshold number of edges have been deleted.

3. The computing device of claim 2, wherein the threshold number of edges comprises five edges.

4. The computing device of any of claims 1-3, wherein the cluster circuitry is further to (i) determine whether the cluster quality metric has decreased in response to deletion of the first edge of the graph and (ii) record a current clustering of the graph and the cluster quality metric in response to a determination the cluster quality metric has not decreased.

5. The computing device of claim 4, wherein the cluster circuitry is further to (i) determine whether sufficient clusterings have been recorded in response to recordation of the

current clustering of the graph and (ii) identify an optimal clustering of the graph based on the cluster quality metric in response to a determination that sufficient clusterings have been recorded.

6. The computing device of claim 4, wherein:

the deletion circuitry is further to reintroduce the first edge into the graph in response to a determination that the cluster quality metric has decreased; and

the centrality circuitry is further to recalculate an approximate vertex centrality weight for each vertex of the graph in response to reintroduction of the first edge into the graph.

7. The computing device of claim 6, wherein:

the deletion circuitry is further to (i) determine whether a backtrack counter exceeds a predetermined backtrack threshold in response to the determination that the cluster quality metric has decreased and (ii) increment the backtrack counter in response to the reintroduction of the first edge into the graph; and

to reintroduce the first edge into the graph further comprises to reintroduce the first edge into the graph in response to a determination that the backtrack counter does not exceed the predetermined backtrack threshold.

8. The computing device of claim 7, wherein to record the cluster quality metric for the current clustering of the graph comprises to record the cluster quality metric for the current clustering of the graph in response to the determination that the cluster quality metric has not decreased or a determination that the backtrack counter exceeds the predetermined backtrack threshold.

9. The computing device of claim 7, wherein the predetermined backtrack threshold comprises ten backtracks.

10. The computing device of any of claims 1-3, wherein the cluster quality metric comprises a modularity metric.

11. The computing device of any of claims 1-3, wherein the cluster quality metric comprises a modified cluster path length.

12. The computing device of claim 11, wherein to compute the modified cluster path length comprises to weight a ratio of average distance between vertices in the graph and average distance between vertices in a cluster by a number of vertices in the cluster.

13. The computing device of claim 11, wherein to compute the modified cluster path length comprises to compute the modified cluster path length as a function of an edge density of the graph.

14. The computing device of any of claims 1-3, wherein to calculate the approximate vertex centrality weight for each vertex of the graph comprises to:

select a pair of distinct vertices from the graph uniformly at random;

compute a set of all shortest paths between the pair of distinct vertices;

select a path from the set of all shortest paths uniformly at random; and

increment the approximate vertex centrality weight of each interior vertex of the path.

15. The computing device of any of claims 1-3, wherein to calculate, based on the approximate vertex centrality weight for each vertex of the graph, the approximate edge centrality value for each edge of the graph comprises to:

select a pair of distinct vertices from the graph with a probability of each vertex equal to the approximate vertex centrality weight of the corresponding vertex;

compute a set of all shortest paths between the pair of distinct vertices;

select a path from the set of all shortest paths uniformly at random; and

increment the approximate edge centrality value of each edge of the path.

16. A method for automatic graph partitioning, the method comprising:

calculating, by a computing device, an approximate vertex centrality weight for each vertex of a graph;

calculating, by the computing device and based on the approximate vertex centrality weight for each vertex of the graph, an approximate edge centrality value for each edge of the graph;

deleting, by the computing device, a first edge of the graph, wherein the first edge connects a first vertex and a second vertex, and wherein the first edge has a largest approximate edge centrality value of the edges of the graph;

determining, by the computing device, whether the first vertex and the second vertex are connected in the graph subsequent to deleting the first edge; and

computing, by the computing device, a cluster quality metric for the graph in response to determining that the first vertex and the second vertex are not connected in the graph subsequent to deleting the first edge.

17. The method of claim 16, further comprising:

determining, by the computing device, whether a threshold number of edges have been deleted in response to determining that the first vertex and the second vertex remain connected in the graph;

deleting, by the computing device, a second edge of the graph in response to determining that the threshold number of edges have not been deleted, wherein the second edge has a largest approximate edge centrality value of the edges remaining in the graph; and

recalculating, by the computing device, an approximate vertex centrality weight for each vertex of the graph in response to determining that the threshold number of edges have been deleted.

18. The method of claim 16, further comprising:

determining, by the computing device, whether the cluster quality metric has decreased in response to deleting the first edge of the graph; and

recording, by the computing device, a current clustering of the graph and the cluster quality metric in response to determining the cluster quality metric has not decreased.

19. The method of claim 18, further comprising:

determining, by the computing device, whether sufficient clusterings have been recorded in response to recording the current clustering of the graph; and

identifying, by the computing device, an optimal clustering of the graph based on the cluster quality metric in response to determining that sufficient clusterings have been recorded.

20. The method of claim 18, further comprising:

determining, by the computing device, whether a backtrack counter exceeds a predetermined backtrack threshold in response to determining that the cluster quality metric has decreased;

reintroducing, by the computing device, the first edge into the graph in response to determining that the cluster quality metric has decreased and in response to determining that the backtrack counter does not exceed the predetermined backtrack threshold;

incrementing, by the computing device, the backtrack counter in response to reintroducing the first edge into the graph; and

recalculating, by the computing device, an approximate vertex centrality weight for each vertex of the graph in response to reintroducing the first edge into the graph;

wherein recording the cluster quality metric for the current clustering of the graph comprises recording the cluster quality metric for the current clustering of the graph in response to determining the cluster quality metric has not decreased or determining that the backtrack counter exceeds the predetermined backtrack threshold.

21. The method of claim 16, wherein computing the cluster quality metric comprises computing a modified cluster path length, wherein computing the modified cluster path length comprises:

weighting a ratio of average distance between vertices in the graph and average distance between vertices in a cluster by a number of vertices in the cluster; and

computing the modified cluster path length as a function of an edge density of the graph.

22. The method of claim 16, wherein:

calculating the approximate vertex centrality weight for each vertex of the graph comprises:

selecting a pair of distinct vertices from the graph uniformly at random;

computing a set of all shortest paths between the pair of distinct vertices;

selecting a path from the set of all shortest paths uniformly at random; and

incrementing the approximate vertex centrality weight of each interior vertex of the path; and

calculating, based on the approximate vertex centrality weight for each vertex of the graph, the approximate edge centrality value for each edge of the graph comprises:

selecting a pair of distinct vertices from the graph with a probability of each vertex equal to the approximate vertex centrality weight of the corresponding vertex;

computing a set of all shortest paths between the pair of distinct vertices;

selecting a path from the set of all shortest paths uniformly at random; and

incrementing the approximate edge centrality value of each edge of the path.

23. A computing device comprising:

a processor; and

a memory having stored therein a plurality of instructions that when executed by the processor cause the computing device to perform the method of any of claims 16-22.

24. One or more machine readable storage media comprising a plurality of instructions stored thereon that in response to being executed result in a computing device performing the method of any of claims 16-22.

25. A computing device comprising means for performing the method of any of claims 16-22.

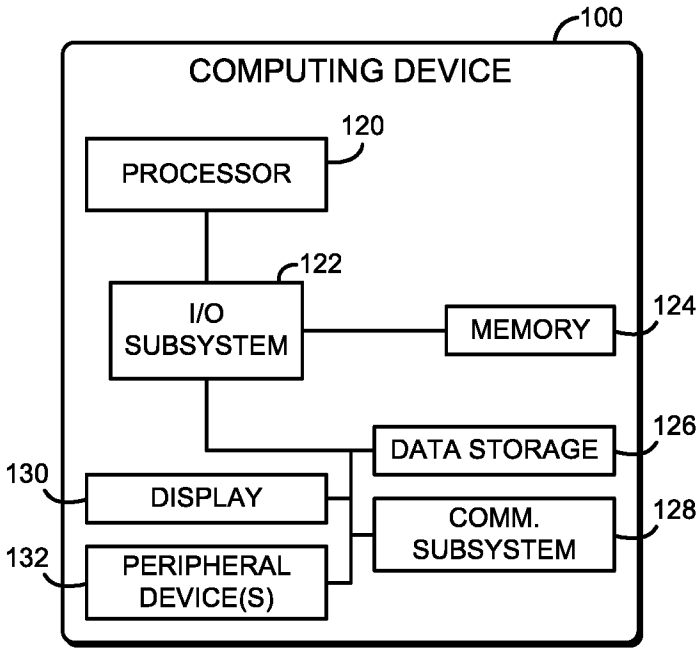
100
COMPUTING DEVICE
120
PROCESSOR
122
I/O SUBSYSTEM
MEMORY
124
DATA STORAGE
126
130
DISPLAY
COMM. SUBSYSTEM
128
132
PERIPHERAL DEVICE(S)

FIG. 1

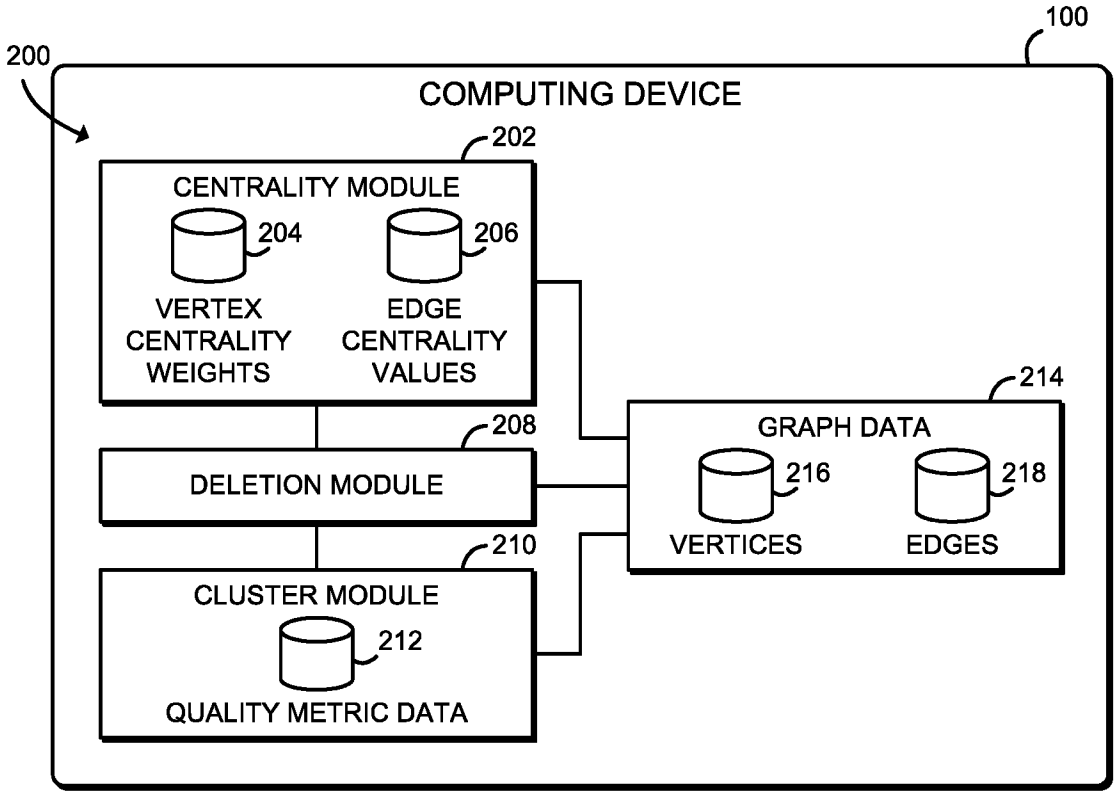
100
200
COMPUTING DEVICE
202
CENTRALITY MODULE
204
206
VERTEX CENTRALITY WEIGHTS
EDGE CENTRALITY VALUES
214
GRAPH DATA
208
DELETION MODULE
216
218
VERTICES
EDGES
210
CLUSTER MODULE
212
QUALITY METRIC DATA

FIG. 2

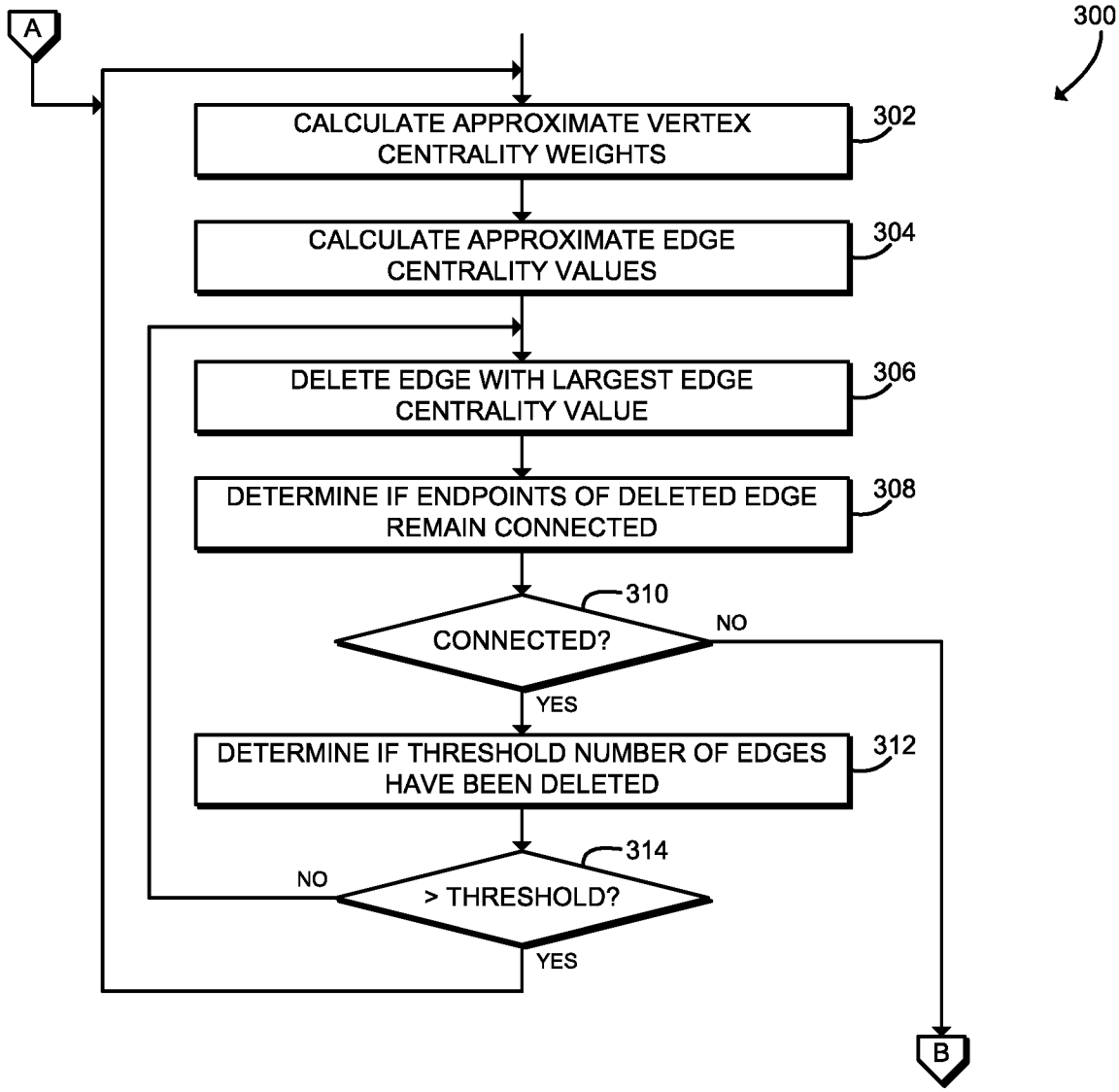
A
300
CALCULATE APPROXIMATE VERTEX CENTRALITY WEIGHTS
302
CALCULATE APPROXIMATE EDGE CENTRALITY VALUES
304
DELETE EDGE WITH LARGEST EDGE CENTRALITY VALUE
306
DETERMINE IF ENDPOINTS OF DELETED EDGE REMAIN CONNECTED
308
310
CONNECTED?
NO
YES
DETERMINE IF THRESHOLD NUMBER OF EDGES HAVE BEEN DELETED
312
314
NO
> THRESHOLD?
YES
B

FIG. 3A

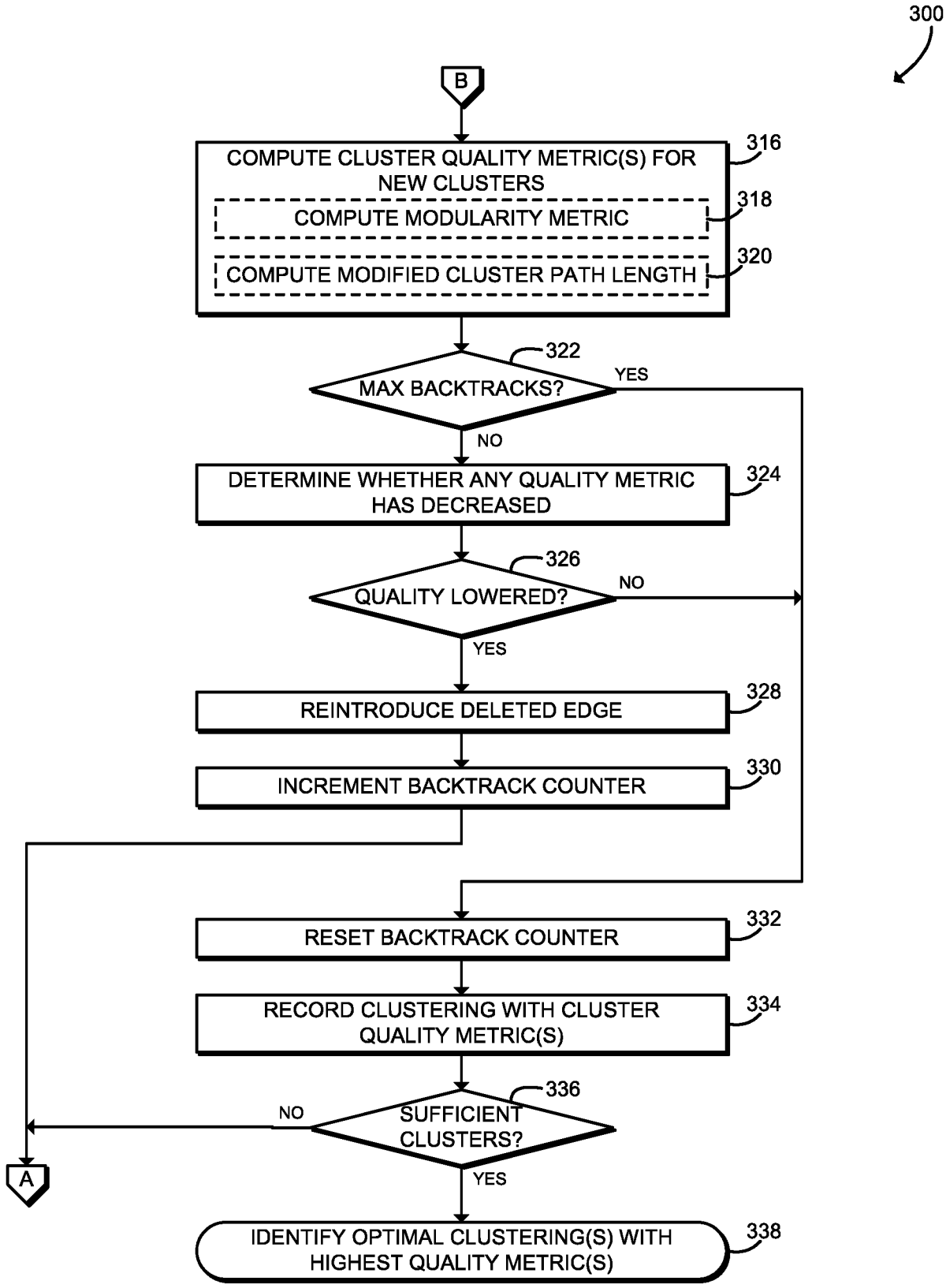
300
B
COMPUTE CLUSTER QUALITY METRIC(S) FOR NEW CLUSTERS
316
COMPUTE MODULARITY METRIC
318
COMPUTE MODIFIED CLUSTER PATH LENGTH
320
322
MAX BACKTRACKS?
YES
NO
DETERMINE WHETHER ANY QUALITY METRIC HAS DECREASED
324
326
QUALITY LOWERED?
NO
YES
REINTRODUCE DELETED EDGE
328
INCREMENT BACKTRACK COUNTER
330
RESET BACKTRACK COUNTER
332
RECORD CLUSTERING WITH CLUSTER QUALITY METRIC(S)
334
336
SUFFICIENT CLUSTERS?
NO
YES
A
IDENTIFY OPTIMAL CLUSTERING(S) WITH HIGHEST QUALITY METRIC(S)
338

FIG. 3B

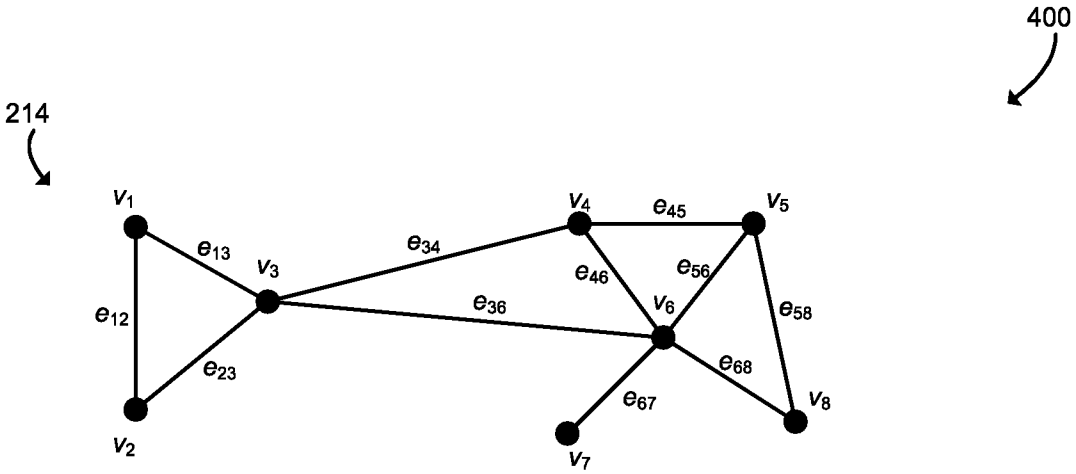

FIG. 4A

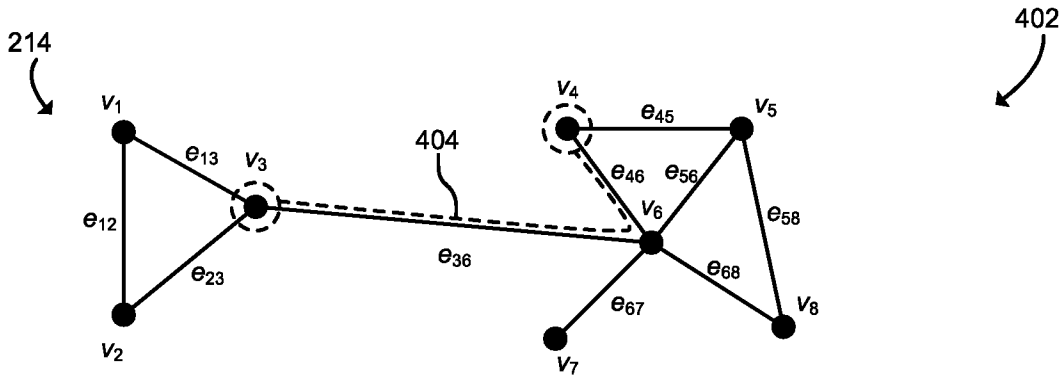

FIG. 4B

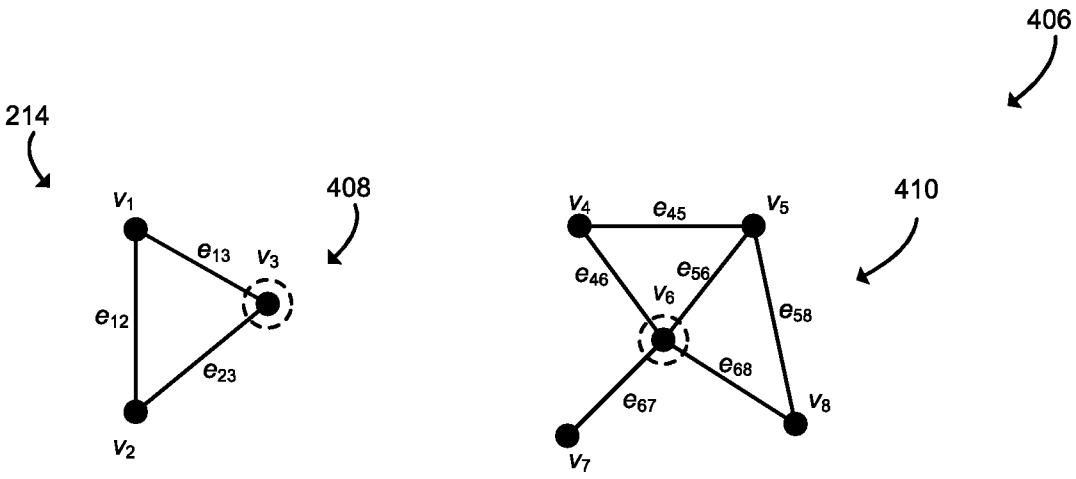

FIG. 4C

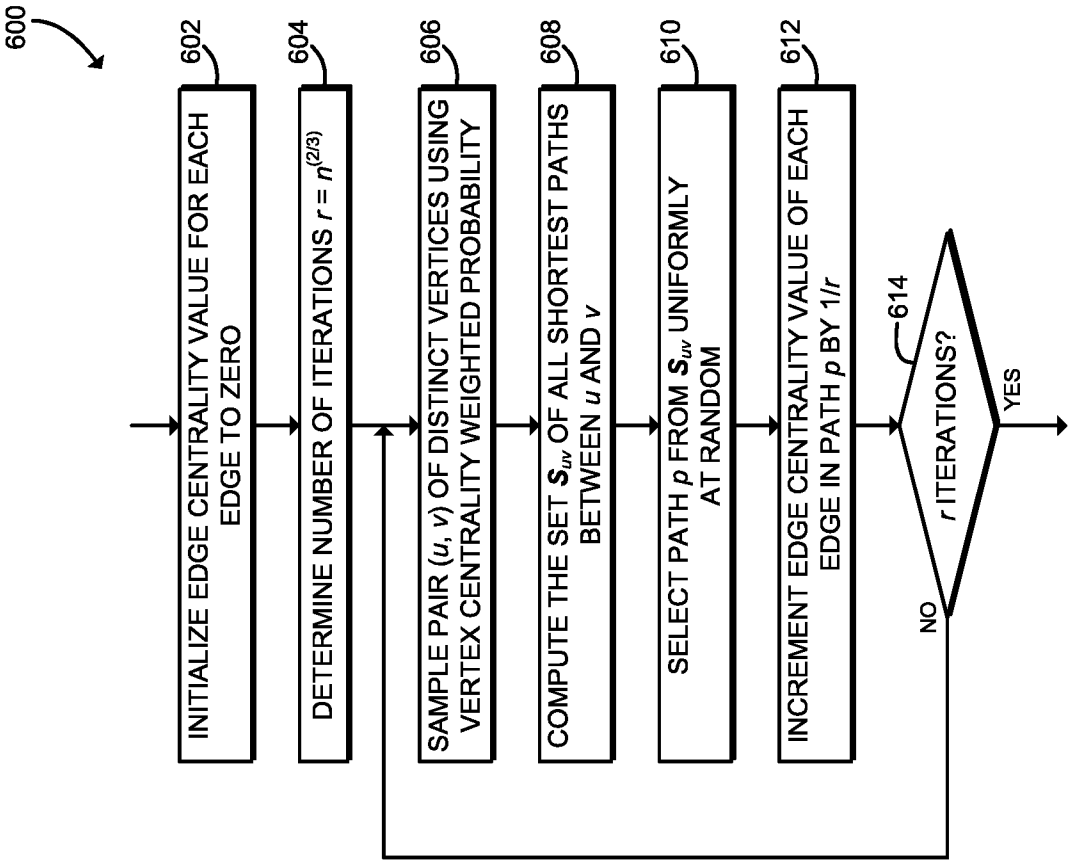
600
602
INITIALIZE EDGE CENTRALITY VALUE FOR EACH EDGE TO ZERO
604
DETERMINE NUMBER OF ITERATIONS $r = n^{(2/3)}$
606
SAMPLE PAIR (u, v) OF DISTINCT VERTICES USING VERTEX CENTRALITY WEIGHTED PROBABILITY
608
COMPUTE THE SET $\mathbf{S}_{uv}$ OF ALL SHORTEST PATHS BETWEEN u AND v
610
SELECT PATH p FROM $\mathbf{S}_{uv}$ UNIFORMLY AT RANDOM
612
INCREMENT EDGE CENTRALITY VALUE OF EACH EDGE IN PATH p BY $1/r$
614
r ITERATIONS?
NO
YES

FIG. 6

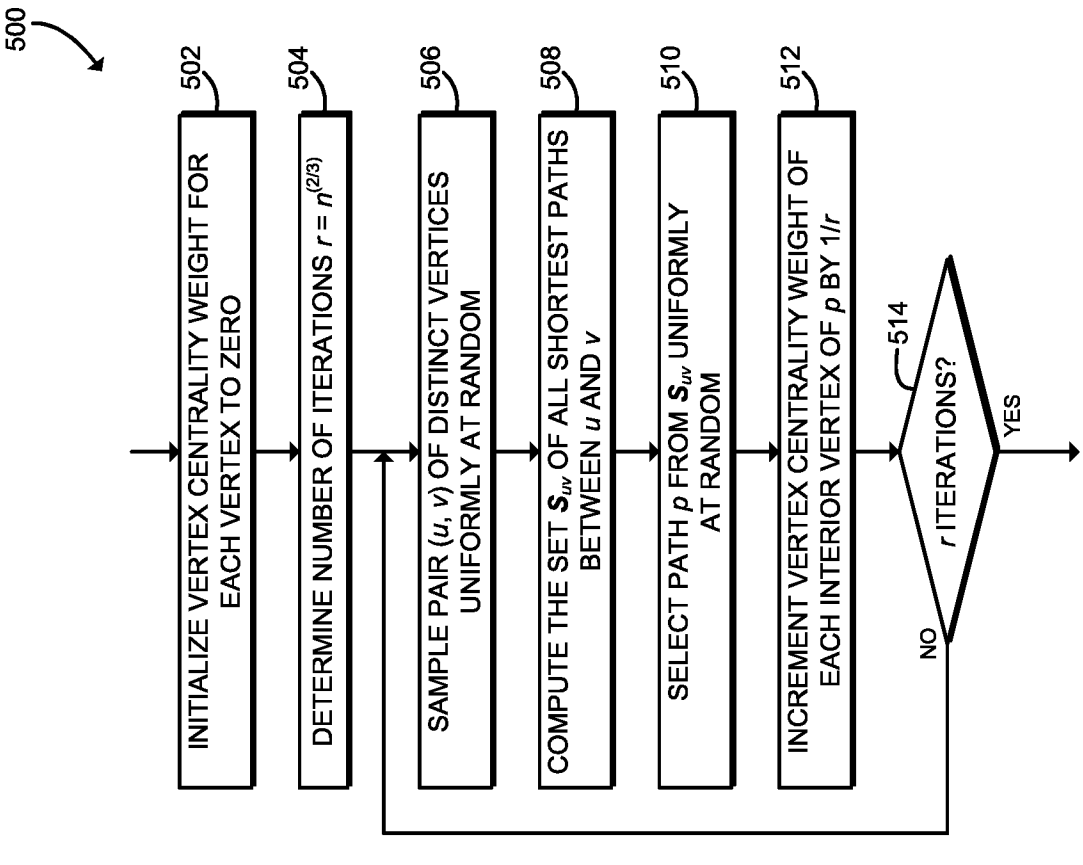
500
502
INITIALIZE VERTEX CENTRALITY WEIGHT FOR EACH VERTEX TO ZERO
504
DETERMINE NUMBER OF ITERATIONS $r = n^{(2/3)}$
506
SAMPLE PAIR (u, v) OF DISTINCT VERTICES UNIFORMLY AT RANDOM
508
COMPUTE THE SET $\mathbf{S}_{uv}$ OF ALL SHORTEST PATHS BETWEEN u AND v
510
SELECT PATH p FROM $\mathbf{S}_{uv}$ UNIFORMLY AT RANDOM
512
INCREMENT VERTEX CENTRALITY WEIGHT OF EACH INTERIOR VERTEX OF p BY $1/r$
514
r ITERATIONS?
NO
YES

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/048595

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30(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 17/30; G06F 9/44; H04L 12/24; G06T 11/20; G01C 21/34; G01C 21/00

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: partitioning, graph, vertex, edge, weight, value, delete, data flow, 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 2015-0067644 A1 (ORACLE INTERNATIONAL CORPORATION) 05 March 2015 See paragraphs [0003]-[0004] and [0021]-[0024]; claims 1 and 8; and figure 4.	1-25
A	US 2014-0320497 A1 (MICROSOFT CORPORATION) 30 October 2014 See paragraphs [0032]-[0033]; claim 1; and figure 3.	1-25
A	US 2013-0268549 A1 (DANIEL DELLING et al.) 10 October 2013 See paragraphs [0029]-[0030]; claim 1; and figure 2.	1-25
A	US 2014-0107921 A1 (MICROSOFT CORPORATION) 17 April 2014 See paragraphs [0036]-[0043]; claim 1; and figure 3.	1-25
A	WO 2015-068083 A1 (TELEFONAKTIEBOLAGET L M ERICSSON (PUBL)) 14 May 2015 See page 15, lines 4-28; claim 1; and figure 4.	1-25

☐ 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
28 November 2016 (28.11.2016)	**28 November 2016 (28.11.2016)**

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
NHO, Ji Myong
Telephone No. +82-42-481-8528

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/048595

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015-0067644 A1	05/03/2015	US 9176732 B2	03/11/2015
US 2014-0320497 A1	30/10/2014	None	
US 2013-0268549 A1	10/10/2013	US 8886573 B2	11/11/2014
US 2014-0107921 A1	17/04/2014	US 9222791 B2	29/12/2015
WO 2015-068083 A1	14/05/2015	EP 3063903 A1 US 2015-0117216 A1 US 9338097 B2	07/09/2016 30/04/2015 10/05/2016