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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 June 2014 (26.06.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/094034 A1

(51) International Patent Classification:
G06F 21/76 (2013.01)

(21) International Application Number:
PCT/AU2013/001430

(22) International Filing Date:
4 December 2013 (04.12.2013)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2012905589 20 December 2012 (20.12.2012) AU

(71) Applicant: **NEWSOUTH INNOVATIONS PTY LIMITED** [AU/AU]; Rupert Myers Building, Gate 14, Barker Street, University of New South Wales, NSW 2052 (AU).

(72) Inventors: **JIANG, Frank**; 20/8 Edmondson St., Campbell, ACT 2612 (AU). **FRATER, Michael**; 32 Westgarth St., O'Connor, ACT 2602 (AU).

(74) Agent: **SPRUSON & FERGUSON**; GPO Box 3898, Sydney, New South Wales 2001 (AU).

(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, LT, 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, 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).

Published:

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

(54) Title: COMPUTER INTRUSION DETECTION

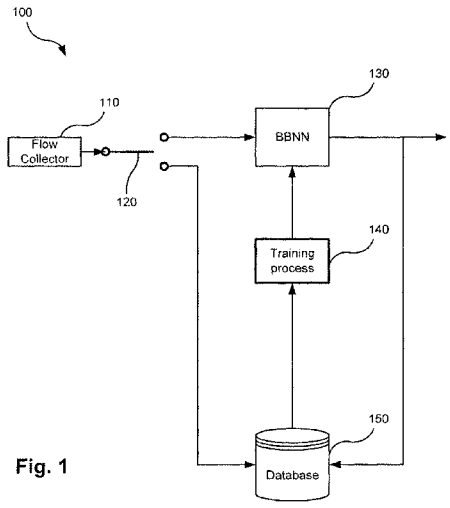

Fig. 1

(57) Abstract: Methods of generating a configuration for block-based neural network (BBNN) structure and intrusion detection systems are disclosed. A flow collector is configured to collect packets destined for computing device and produce input vector for each detected flow of packets. A BBNN structure is developed and configured to label each input vector as either normal or intrusion. A FPGA-based hardware system is provided in which BBNN structure is embedded to present a flexible structure for identifications of time-varying nature of intruders as well as new and unknown intrusions. A database is configured to store labelled input vectors. A computing device is configured to execute real-time detection process, wherein a pre-training process is configured to process the labelled vectors to generate a configuration for BBNN structure. A detection device operates with stable convergence rates under a flexible structure in which a high detection rate is maintained with low false alarm rate.

WO 2014/094034 A1

COMPUTER INTRUSION DETECTION

Related Application

[0001] The present patent application claims the benefit of the earlier filing date of Australian Provisional Patent Application No. 2012905589 filed 20 December 2012 of the same title, which is incorporated herein by reference in its entirety.

Technical Field

[0002] The present disclosure relates to computer security and in particular to detecting intrusion into computing devices using reconfigurable hardware.

Background

[0003] Large-scale distributed computing infrastructure is presenting new challenges to conventional intrusion detection, prevention, and self-healing security systems. For example, the past few years have seen over 600 million dollars in losses accrue annually across Australian businesses due to computer security incidents. As it is impossible to completely prevent computer attacks, intrusion detection systems (IDSs) are designed to minimize the damage caused by potential attackers.

[0004] Traditionally, IDSs include network-based and host-based implementations. They can also be roughly divided into two primary categories: rate-based detection, and content-based detection (i.e., signature and anomaly-based). A rate-based IDS blocks the suspicious traffic whenever a threshold, e.g. of number of packets or connections in a given time interval, is reached. A content-based IDS is triggered by anomalous behaviour of the networks (e.g., port scanning), by matching a signature of the behaviour with that of known anomalies, such as Sasser, Blaster, MyDoom, etc. Content-based IDSs are generally classified into the following categories: (1) statistical features and correlation analysis (signal processing approaches); (2) artificial intelligence-based, rule-based, and agent-based approaches; and (3) data mining-based approaches. However, in addition to the very limited public IDS datasets, current IDS systems in use are primarily facing the following two difficulties: (1) comparatively high error rate of detection, especially for "new" (unknown or evolving) attacks; and (2) low speed of detection, especially in the large-scale real-time environment, such as core networks.

[0005] A purely software-implemented IDS can adapt itself effectively to new attacks. However, its detection speed is limited. Many modern intrusion detection processes run too slowly in software to have any practical value, and cannot meet the requirements in a large-scale distributed environment in future computing scenarios such as cloud computing, where enormous amounts of data are merged into a centralized infrastructure. In contrast, a purely hardware-implemented IDS will provide a relatively high detection speed in comparison with existing software IDSs, but is not able to detect new attacks.

Summary

[0006] The aspects of the present invention seek to overcome, or at least ameliorate, one or more of the above disadvantages of existing arrangements.

[0007] Disclosed is an intrusion detection system that is hardware-cored with adaptive capability to cope with both high rates of incoming data and evolving attacks. The disclosed IDS uses a high-sampling-rate-enabled field-programmable gate array (FPGA), on which is implemented an adaptive Block-Based Neural Network (BBNN) engine to perform attack-related pattern recognition. The internal block structure of the BBNN matches well with the reconfigurable logic gate array structure of FPGA hardware. The BBNN is software-driven to allow dynamic reconfiguration. The configuration of the BBNN is determined during an offline training stage by an optimisation process referred to as Spiral Particle Swarm Optimisation with Wavelet Mutation (SPSOWM). The disclosed IDS has higher speed than purely software implementations and the ability to adapt to unknown and / or evolving attacks with lower false alarm rates and higher detection rates in a real-time environment than conventional approaches.

[0008] In accordance with an aspect of the invention, there is provided an intrusion detection system. The system comprises: a flow collector configured to collect packets destined for a computing device and produce an input vector for each detected flow of packets; a block-based neural network structure developed and configured to label each input vector as either normal or an intrusion, a FPGA-based hardware system in which the block-based neural network structure is embedded to present a flexible structure for identifications of time-varying nature of intruders as well as new and unknown intrusions; a database configured to store labelled input vectors; a computing device configured to execute a real-time detection process, wherein a pre-training process is configured to process the labelled input vectors to generate a configuration for the

block-based neural network structure; and a detection device that operates with stable convergence rates under a flexible structure in which a high detection rate is maintained with low false alarm rate.

[0009] In accordance with an aspect of the invention, there is provided a method of generating a configuration for a block-based neural network structure. The method comprises:

initializing a position and a velocity of each component of each particle in a particle swarm, the weighted structure encoded in a 2-D array representation as the configuration of a block-based neural network structure; developing new movements of particles to solve a stagnation problem; determining whether a termination condition is met by the particle swarm, and if so, returning the configuration represented by a global best position of all the particles in the swarm; otherwise updating the velocity of each component of each particle using a velocity update rule; updating the position of each component of each particle using a position update rule and the corresponding updated component velocity; mutating at least one particle using a mutation rule; reproducing the swarm using the mutated particles; updating the global best position of all the particles in the reproduced swarm; and repeating the foregoing steps until the termination condition is met; wherein the velocity update rule uses as a target best position for the particle a best position for a particle other than the current particle being updated.

Brief Description of Drawings

[00010] At least one embodiment of the invention is described below with reference to the drawings, in which:

[00011] Fig. 1 is a block diagram of an intrusion detection system according to one embodiment;

[00012] Fig. 2 is a block diagram illustrating a block-based neural network used as used in the system of Fig. 1;

[00013] Fig. 3 is a block diagram illustrating the four possible structures of each block in the block-based neural network of Fig. 2;

[00014] Figs. 4A and 4B form a schematic block diagram of a general purpose computing device on which the training process in the system of Fig. 1 may be implemented;

[00015] Fig. 5 is a flow chart illustrating a particle swarm optimisation (PSO) algorithm with mutation that may be used to implement the training process 140 of Fig. 1; and

[00016] Figs. 6A and 6B illustrate the difference between typical particle paths in standard PSO and SPSOWM.

Description of Embodiments

[00017] Fig. 1 is a block diagram of an intrusion detection IDS 100 according to one embodiment. Packets arrive at the flow collector 110 from one or more routers (not shown). The flow collector 110 analyses the packets to produce an input vector for each flow in the manner described below. In the seven-layer OSI model of computer networking, 'packet' strictly refers to a data unit at layer 3. As for the "flow", a sequence of packets sent from a particular source to a particular unicast, anycast, or multicast destination that the source desires to label is a flow. A flow comprises all packets in a specific transport connection or a media stream. During the training phase, the automatic switch 120 is in the lower position. Each vector has a corresponding label (intrusion or normal) with which it is stored in the database 150. The labels on the vectors may be obtained from of the input packets or may be manually applied by an operator during the training phase. A training process 140, described in detail below, processes the labelled vectors in the database 150 to generate a configuration for a block-based neural network (BBNN) 130, implemented in hardware. In one implementation, the BBNN 130 is implemented using a field programmable gate array (FPGA). In some implementations, the training process 140 and the database 150 are implemented on the same physical device. In other implementations, the training process 140 and the database 150 are implemented on separate physical devices.

[00018] During the operation phase, the switch 120 is in the upper position, so that incoming vectors pass into the BBNN 130, which processes each vector to generate a label (intrusion or normal) for the corresponding flow. The generated labels are passed to subsequent stages for possible action. These actions refer to the general detection or re-examination, etc. The labelled vectors are also stored in the database 150 for subsequent training phases.

[00019] The flow collector 110 is implemented in hardware to enable it to operate at the required real-time rate. Operation processing time reaches milliseconds level within the acceptable amount of data input. The processing carried out by the flow collector 110 depends on the protocol followed by the input packets. In one implementation, the input packets are NetFlow packets, which include among their fields Source and Destination IP addresses, Source and Destination Ports, and a timestamp. The flow collector 110 parses the fields of each input packet and generates, for each flow, a vector comprising the four features shown in Table 1:

Name of feature	**Description**
Packets	The number of packets in the flow
Bytes	The number of bytes in the flow
Duration	The number of seconds from the start time of the flow to the end time of the flow
Flags	The decimal value of the TCP flags (represented in hexadecimal)

Table 1: Flow Features used by IDS 100

[00020] The flow collector 110 scales each feature value x to the range [-1, 1] using the following equation:

$$x \to \frac{2(x - x_{\min})}{x_{\max} - x_{\min}} - 1 \qquad (1)$$

[00021] where $x_{\max}$ and $x_{\min}$ and the maximum and minimum values of the feature value x over the whole dataset.

[00022] The flow collector 110 then forwards the scaled vector to the database 150 or the BBNN 130 for storage or processing.

[00023] In one implementation, the BBNN 130 is implemented using the Cyclone III FPGA Starter Board manufactured by the Altera Corporation. This model has a 50 MHz clock, 32 Mbytes of SRAM, 4 LEDs, and a USB programming interface.

[00024] Fig. 2 is a block diagram illustrating a BBNN 200 used as the BBNN 130 in the system 100 of Fig. 1. The BBNN 200 is a two-dimensional array of interconnected blocks, e.g. 220. The input 210 to the BBNN 200 is a vector **x** of n components, or features, x_j. The BBNN 200 comprises m rows or layers, each comprising n blocks, so the blocks 220 are therefore labelled as B_{ij} where i runs from 1 to m and j runs from 1 to n. The output 230 of the BBNN 200 is a vector **y** of n components, each labelled as y_j. Vector **y** consists of a set of y_i. It is a function of summation of weighted inputs and a bias. Also as shown in Fig. 2, the rightmost node has arbitrarily chosen as the output. All the redundant output nodes are marked as *. This rightmost node output is denoted as 1 (intrusion) or 0 (no intrusion).

[00025] Each individual block 220 is a basic signal processing unit (neuron) having four variable input/output nodes labelled 1 (left), 2 (right), 3 (top), and 4 (bottom). Node 1 is always an input node and node 2 is always an output node, so signal flow is always from left to right, whereas nodes 3 and 4 can be either input or output, so there are four possible configurations for each block 220. For example, in Fig. 2, the top node 3 and the bottom node 4 of the block B_{22} (220) are both outputs.

[00026] Fig. 3 is a block diagram illustrating the four possible configurations 310, 320, 330, and 340 of each block 220 in the BBNN 200 of Fig. 2. In the configuration 310 (type 1), the top node 3 is an input and the bottom node 4 is an output. In the configuration 320 (type 2), the top node 3 and the bottom node 4 are both inputs. In the configuration 330 (type 3), the top node 3 and the bottom node 4 are both outputs. Finally, in the configuration 340 (type 4), the top node 3 is an output and the bottom node 4 is an input. The inputs to a block are labelled as u_i, where i is 1, 3, or 4, whereas the outputs are labelled as v_j, where j is 2, 3, or 4. The output of each configuration is computed by the summation of its weighted inputs and a bias term b_j:

$$v_j = \sum_{i=1}^{4} w_{ij} u_i + b_j \qquad (2)$$

where w_{ij} is the weight from input i to output j. Because node 1 is always an input and node 2 always an output, the weights w_{11}, w_{21}, w_{31}, w_{41}, w_{22}, w_{23}, w_{24}, w_{33}, and w_{44} are identically zero.

A type 1 configuration 310 has four non-zero weights (w_{12}, w_{14}, w_{32}, and w_{34}) and two non-zero biases (b_2 and b_4). A type 2 configuration 320 has three non-zero weights (w_{12}, w_{23}, and w_{24}) and one non-zero bias (b_2). A type 3 configuration 330 has three non-zero weights (w_{12}, w_{13}, and w_{14}) and three non-zero biases (b_2, b_3, and b_4). A type 4 configuration 340 has four non-zero weights (w_{12}, w_{13}, w_{42}, and w_{43},) and two non-zero biases (b_2 and b_3). Therefore, each block 220 may be completely characterised by two bits (indicating one of the four configurations) and at most six scalar parameters.

[00027] The inputs x_j to the BBNN 200 may be identified with the top inputs u_3 to the first layer blocks B_{1n}, which must therefore be of type 1 or type 2. The connection directions between the blocks are flexible, here in Fig 2, the directions always from left hand side to right hand side; some connections record can be from right to left. Therefore, the structure constraints are minimised. The outputs y_j of the BBNN 200 may be identified with the bottom outputs v_4 of the m-th layer blocks B_{mn}, which must therefore be of type 1 or type 3. The BBNN 130 may therefore be characterised by $10mn$ parameters, each representable by at least four bits, and $2mn$ "structure" bits.

[00028] Figs. 4A and 4B collectively form a schematic block diagram of a general purpose computing device 400, upon which the training process 140 and / or the database 150 may be implemented.

[00029] As seen in Fig. 4A, the computing device 400 is formed by a computer module 401, input devices such as a keyboard 402, a pointer device 403, and a microphone 480, and output devices including a printer 415, a display device 414 and loudspeakers 417. An external Modulator-Demodulator (Modem) transceiver device 416 may be used by the computer module 401 for communicating to and from a communications network 420 via a connection 421. The network 420 may be a wide-area network (WAN), such as the Internet or a private WAN. Where the connection 421 is a telephone line, the modem 416 may be a "dial-up" modem. Alternatively, where the connection 421 is a high capacity (e.g. cable) connection, the modem 416 may be a broadband modem. A wireless modem may also be used for wireless connection to the network 420.

[00030] The computer module 401 typically includes at least one processor unit 405, and a memory unit 406 for example formed from semiconductor random access memory (RAM) and

semiconductor read only memory (ROM). The module 401 also includes an number of input/output (I/O) interfaces including an audio-video interface 407 that couples to the video display 414, loudspeakers 417 and microphone 480, an I/O interface 413 for the keyboard 402 and pointer device 403, and an interface 408 for the external modem 416 and printer 415. In some implementations, the modem 416 may be incorporated within the computer module 401, for example within the interface 408. The computer module 401 also has a local network interface 411 which, via a connection 423, permits coupling of the computing device 400 to a local computer network 422, known as a Local Area Network (LAN). As also illustrated, the local network 422 may also couple to the wide network 420 via a connection 424, which would typically include a so-called "firewall" device or device of similar functionality. The interface 411 may be formed by an Ethernet™ circuit card, a Bluetooth™ wireless arrangement or an IEEE 802.11 wireless arrangement.

[00031] The interfaces 408 and 413 may afford either or both of serial and parallel connectivity, the former typically being implemented according to the Universal Serial Bus (USB) standards and having corresponding USB connectors (not illustrated). In particular, the computing device 400 may interface with the BBNN 130 over such a USB interface 408.

[00032] Storage devices 409 are provided and typically include a hard disk drive (HDD) 410. Other storage devices such as a floppy disk drive and a magnetic tape drive (not illustrated) may also be used. A reader 412 is typically provided to interface with an external non-volatile source of data. A portable computer readable storage device 425, such as optical disks (e.g. CD-ROM, DVD), USB-RAM, and floppy disks for example may then be used as appropriate sources of data to the system 400. In particular, the database 150 may be implemented in the storage device 409 or the portable computer readable storage device 425.

[00033] The components 405 to 413 of the computer module 401 typically communicate via an interconnected bus 404 and in a manner which results in a conventional mode of operation of the computing device 400 known to those in the relevant art. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations, Apple Mac™ or computing devices evolved therefrom.

[00034] The training process 140 may be implemented using the computing device 400 as one or more software programs 433 executable within the computing device 400. In particular, with

reference to Fig. 4B, the steps of the training process 140 are effected by instructions 431 in the software 433 that are carried out within the computing device 400. The software instructions 431 may be formed as one or more code modules, each for performing one or more particular tasks. The software may also be divided into two separate parts, in which a first part and the corresponding code modules performs the training process 140 and a second part and the corresponding code modules manage a user interface between the first part and the user.

[00035] The software 433 is generally loaded into the computing device 400 from a computer readable medium, and is then typically stored in the HDD 410, as illustrated in Fig. 4A, or the memory 406, after which the software 433 can be executed by the computing device 400. In some instances, the programs 433 may be supplied to the user encoded on one or more storage media 425 and read via the corresponding reader 412 prior to storage in the memory 410 or 406. Computer readable storage media refers to any non-transitory tangible storage medium that participates in providing instructions and/or data to the computing device 400 for execution and/or processing. Examples of such storage media include floppy disks, magnetic tape, CD-ROM, DVD, a hard disk drive, a ROM or integrated circuit, USB memory, a magneto-optical disk, semiconductor memory, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external to the computer module 401. A computer readable storage medium having such software or computer program recorded on it is a computer program product. The use of such a computer program product in the computer module 401 effects an apparatus for training the BBNN 130.

[00036] Alternatively the software 433 may be read by the computing device 400 from the networks 420 or 422 or loaded into the computing device 400 from other computer readable media. Examples of transitory or non-tangible computer readable transmission media that may also participate in the provision of software programs, instructions and/or data to the computer module 401 include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

[00037] The second part of the software 433 and the corresponding code modules mentioned above may be executed to implement one or more graphical user interfaces (GUIs) to be rendered or otherwise represented upon the display 414. Through manipulation of typically the keyboard 402 and the mouse 403, a user of the computing device 400 and the software 433 may

manipulate the interface in a functionally adaptable manner to provide controlling commands and/or input to the software associated with the GUI(s). Other forms of functionally adaptable user interfaces may also be implemented, such as an audio interface utilizing speech prompts output via the loudspeakers 417 and user voice commands input via the microphone 480.

[00038] Fig. 4B is a detailed schematic block diagram of the processor 405 and a "memory" 434. The memory 434 represents a logical aggregation of all the memory devices (including the HDD 410 and semiconductor memory 406) that can be accessed by the computer module 401 in Fig. 4A.

[00039] When the computer module 401 is initially powered up, a power-on self-test (POST) program 450 executes. The POST program 450 is typically stored in a ROM 449 of the semiconductor memory 406. A program permanently stored in a hardware device such as the ROM 449 is sometimes referred to as firmware. The POST program 450 examines hardware within the computer module 401 to ensure proper functioning, and typically checks the processor 405, the memory (409, 406), and a basic input-output systems software (BIOS) module 451, also typically stored in the ROM 449, for correct operation. Once the POST program 450 has run successfully, the BIOS 451 activates the hard disk drive 410. Activation of the hard disk drive 410 causes a bootstrap loader program 452 that is resident on the hard disk drive 410 to execute via the processor 405. This loads an operating system 453 into the RAM memory 406 upon which the operating system 453 commences operation. The operating system 453 is a system level application, executable by the processor 405, to fulfill various high level functions, including processor management, memory management, device management, storage management, software application interface, and generic user interface.

[00040] The operating system 453 manages the memory (409, 406) in order to ensure that each process or application running on the computer module 401 has sufficient memory in which to execute without colliding with memory allocated to another process. Furthermore, the different types of memory available in the system 400 must be used properly so that each process can run effectively. Accordingly, the aggregated memory 434 is not intended to illustrate how particular segments of memory are allocated (unless otherwise stated), but rather to provide a general view of the memory accessible by the computing device 400 and how such is used.

[00041] The processor 405 includes a number of functional modules including a control unit 439, an arithmetic logic unit (ALU) 440, and a local or internal memory 448, sometimes called a cache memory. The cache memory 448 typically includes a number of storage registers 444 - 446 in a register section. One or more internal buses 441 functionally interconnect these functional modules. The processor 405 typically also has one or more interfaces 442 for communicating with external devices via the system bus 404, using a connection 418.

[00042] The program 433 includes a sequence of instructions 431 that may include conditional branch and loop instructions. The program 433 may also include data 432 which is used in execution of the program 433. The instructions 431 and the data 432 are stored in memory locations 428-430 and 435-437 respectively. Depending upon the relative size of the instructions 431 and the memory locations 428-430, a particular instruction may be stored in a single memory location as depicted by the instruction shown in the memory location 430. Alternately, an instruction may be segmented into a number of parts each of which is stored in a separate memory location, as depicted by the instruction segments shown in the memory locations 428-429.

[00043] In general, the processor 405 is given a set of instructions which are executed therein. The processor 405 then waits for a subsequent input, to which it reacts to by executing another set of instructions. Each input may be provided from one or more of a number of sources, including data generated by one or more of the input devices 402, 403, data received from an external source across one of the networks 420, 422, data retrieved from one of the storage devices 406, 409 or data retrieved from a storage medium 425 inserted into the corresponding reader 412. The execution of a set of the instructions may in some cases result in output of data. Execution may also involve storing data or variables to the memory 434.

[00044] The training process 140 use input variables 454, that are stored in the memory 434 in corresponding memory locations 455-457. The training process 140 produces output variables 461, that are stored in the memory 434 in corresponding memory locations 462-464. Intermediate variables may be stored in memory locations 459, 460, 466 and 467.

[00045] The register section 444-446, the arithmetic logic unit (ALU) 440, and the control unit 439 of the processor 405 work together to perform sequences of micro-operations needed to

perform "fetch, decode, and execute" cycles for every instruction in the instruction set making up the program 433. Each fetch, decode, and execute cycle comprises:

(a) a fetch operation, which fetches or reads an instruction 431 from a memory location 428;

(b) a decode operation in which the control unit 439 determines which instruction has been fetched; and

(c) an execute operation in which the control unit 439 and/or the ALU 440 execute the instruction.

[00046] Thereafter, a further fetch, decode, and execute cycle for the next instruction may be executed. Similarly, a store cycle may be performed by which the control unit 439 stores or writes a value to a memory location 432.

[00047] Each step or sub-process in the training process 140 is associated with one or more segments of the program 433, and is performed by the register section 444-447, the ALU 440, and the control unit 439 in the processor 405 working together to perform the fetch, decode, and execute cycles for every instruction in the instruction set for the noted segments of the program 433.

[00048] The training process 140 may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of the training process 140. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

[00049] The training process 140 comprises two tasks: structure and parameter optimisation. Structure optimisation corresponds to the determination of the type of each block 220 in the BBNN 130. Parameter optimisation corresponds to determining the weights and biases in each block 220. In one implementation, the training process 140 is a genetic algorithm (GA). The genetic algorithm searches for the global optimum of structure and parameters for the BBNN 130, within permitted combinations of structure and parameters, based on a given fitness function of the structure and parameters. The fitness function is to achieve the optimized detection rate while the false positive rate is kept at acceptable value. The GA is used to find a maximum value of the fitness function, which occurs at an "optimum" set of structures and parameters of the BBNN 130. However, as far as the model-free neural network structure is

concerned, the relative slowness of a training process 140 implemented by a GA constrains the ability of such an IDS 100 to cope with a real-time application. In other implementations, the training process 140 is a Particle Swarm Optimisation (PSO) algorithm.

[00050] To describe PSO algorithms, the following notation is used. $X(t)$ denotes a particle swarm containing P particles at the t-th iteration. Each particle is represented by a vector $\mathbf{x}^p(t)$, where $p = 1, \ldots, P$, containing κ components denoted as $x_j^p(t)$ for $j = 1, \ldots, \kappa$. The value of each component $x_j^p(t)$ of $\mathbf{x}^p(t)$ represents the position of the particle indexed by p along the j-th coordinate axis. Each particle position $\mathbf{x}^p(t)$ represents a candidate optimising solution for the predetermined objective function denoted as $f(\mathbf{x})$. An objective function F may also be evaluated on the entire particle swarm $X(t)$, based on the values of the objective function $f(\mathbf{x}^p(t))$ for $p = 1, \ldots, P$. The objective function depicts the satisfied intrusion detection rate performance for BBNN, given a condition that false alarm rate is less than an acceptable value. The aim of a PSO algorithm is to optimise the fitness function $F(X(t))$ by iterative refinement of the particle positions $\mathbf{x}^p(t)$. In the case where the objective function f is a fitness function, "optimisation" means maximisation. In the case where the objective function f is a cost function, "optimisation" means minimisation.

[00051] In PSO algorithms, the positions $\mathbf{x}^p(t)$ of the particles in the swarm $X(t)$ are first initialised and the iteration number t is set to zero. The objective function $F(X(t))$ is evaluated on the swarm $X(t)$. If a termination condition, typically dependent on the value of the objective function $F(X(t))$, is not met, the swarm $X(t)$ is updated from iteration t-1 to iteration t by updating the velocity $\mathbf{v}^p(t)$ of each particle according to a velocity update rule, and then updating the position $\mathbf{x}^p(t)$ of each particle according to a position update rule using the updated velocity $\mathbf{v}^p(t)$. A new swarm is then "reproduced" from the updated particle positions $\mathbf{x}^p(t)$. A "best" position $\bar{\mathbf{x}}^p$ of each particle and a global "best" position $\bar{\mathbf{x}}$ of the reproduced swarm are then updated. The algorithm then returns to test the termination condition on the reproduced swarm; if satisfied, the global "best" position $\bar{\mathbf{x}}$ is returned; otherwise, a further update is performed.

[00052] In PSO algorithms with mutation, the swarm $X(t)$ is "mutated" according to a mutation rule after the updating steps, and the new swarm is reproduced from the mutated swarm.

[00053] Fig. 5 is a flow chart illustrating a PSO-based method 500 that may be used to implement the training process 140 of Fig. 1. In the implementation in which the training process 140 is implemented as software in the computing device 400, the method 500 is carried out by the processor 405 in concert with the software 433.

[00054] In the method 500, the particle positions $\mathbf{x}^p(t)$ are binary-valued, and the binary values of each component $x_j^p(t)$ of each particle position encode the structure and parameters of the BBNN 200. Each block 220 in the BBNN 200 is encoded by six parameters and two structure bits, as mentioned above. Each parameter is encoded by B bits. Since there are mn blocks 220 in the BBNN 200, $mn(6B+2)$ bits are required in total to encode the structure and parameters of the BBNN 200. Each particle in the swarm $X(t)$ therefore comprises $\kappa = mn(6B+4)$ components that are grouped into mn groups of $6B+2$, each encoding a block 220 in the BBNN 200. Within each group, 6 subgroups of B bits each encode the six parameters of the block 220, while the remaining two bits encode the structure of the block 220. (For a block of type 2, e.g. 340 in Fig. 3, only four parameters are needed, so the fifth and sixth subgroups are ignored.) The two structure bits indicate the directions of the vertical signals indexed by 3 and 4 in Fig. 3: 0 indicates downward flow, while 1 indicates upward flow.

[00055] The method 500 starts at step 510, where the iteration number t is initialised to zero. The position $\mathbf{x}^p(t)$ and velocity $\mathbf{v}^p(t)$ of each particle in the swarm $X(t)$ are initialised, to most suitable initial values, such as zero, at the following step 520. Step 520 also initialises the "best" position $\bar{\mathbf{x}}^p$ of each particle to the corresponding initialised value $\mathbf{x}^p(0)$, and the global "best" position $\bar{\mathbf{x}}$ of the entire swarm $X(t)$ to the position $\bar{\mathbf{x}}^p$ that optimises the objective function $f(\bar{\mathbf{x}}^p)$ over all p.

[00056] Step 530 follows, at which the objective function $F(X(t))$ is evaluated on the particle swarm $X(t)$ for the current iteration. Step 540 then tests whether the termination condition is met. For example, the termination condition might involve evaluating f(x(t)); if the predefined ratio is achieved or if the predefined iteration number is reached, then terminate the process. If so ("Y"), the method 500 concludes at step 590, in which a best global position or velocity is returned. Otherwise ("N"), the method 500 proceeds to step 545, at which the iteration number t is incremented. At the next step 550, the velocity $\mathbf{v}^p(t)$ of each particle is updated according to a velocity update rule that typically involves one or more of the current best position $\bar{\mathbf{x}}^p$, the

current global best position $\bar{\mathbf{x}}$, and the current position $\mathbf{x}^p(t-1)$. As part of the velocity update rule, step 550 "clips" the updated velocity $v_j^p(t)$ to remain in the range [$-v_{max}$, v_{max}] for a predetermined maximum velocity v_{max}, where v_{max} is typically set to 10% - 20% of the dynamic range of corresponding component.

[00057] The method 500 then proceeds to step 555, at which the position $\mathbf{x}^p(t)$ of each particle is updated using a position update rule involving the updated velocity $\mathbf{v}^p(t)$. The velocity update rule used at step 550 and the position update rule used at step 555 are described in detail below.

[00058] At the following step 560, the position $\mathbf{x}^p(t)$ of each particle in the swarm $X(t)$ is mutated according to a mutation rule. The mutation rule used at step 560 is described in detail below. The method 500 then at step 570 reproduces a new swarm $X(t)$ from the mutated swarm. The mutation rules are applied in a normal way

[00059] Step 580 follows, at which the best position $\bar{\mathbf{x}}^p$ of each particle and the global best position $\bar{\mathbf{x}}$ of the reproduced swarm $X(t)$ are updated. Step 580 updates $\bar{\mathbf{x}}^p$ to the current particle position $\mathbf{x}^p(t)$ if the objective function $f(\mathbf{x}^p(t))$ is "more optimal" than the objective function $f(\bar{\mathbf{x}}^p)$ of the current best position $\bar{\mathbf{x}}^p$, and updates $\bar{\mathbf{x}}$ to one of the best positions $\bar{\mathbf{x}}^p$ if the objective function $f(\bar{\mathbf{x}}^p)$ for some p is "more optimal" than the objective function $f(\bar{\mathbf{x}})$. That is, the case where f is a fitness function to be maximised,

$$\bar{\mathbf{x}}^p \rightarrow \begin{cases} \mathbf{x}^p(t), f(\mathbf{x}^p(t)) > f(\bar{\mathbf{x}}^p) \\ \bar{\mathbf{x}}^p, \text{otherwise} \end{cases} \tag{3}$$

$$\bar{\mathbf{x}} \rightarrow \begin{cases} \bar{\mathbf{x}}^p, f(\bar{\mathbf{x}}^p) > f(\bar{\mathbf{x}}) \text{ for some } p \in [1, P] \\ \bar{\mathbf{x}}, \text{otherwise} \end{cases} \tag{4}$$

[00060] In the case where f is a cost function to be minimised, equations (3) and (4) are still used, but with the inequalities reversed.

[00061] After step 580, the method 500 returns to step 530 for another iteration, if needed.

[00062] One example of a velocity update rule is given by:

$$v_j^p(t) = k\left(wv_j^p(t-1) + \phi_1 r_1\left(\bar{x}_j^p - x_j^p(t-1)\right) + \phi_2 r_2\left(\bar{x}_j - x_j^p(t-1)\right)\right) \quad (5)$$

[00063] where:

- w is an inertia constant;
- ϕ_1 and ϕ_2 are acceleration constants;
- r_1 and r_2 are random numbers between 0 and 1; and
- k is a constriction factor that is derived from stability analysis of equation (5) as

$$k = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \quad (6)$$

[00064] where $\phi = \phi_1 + \phi_2$ (ϕ_1 and ϕ_2 are chosen such that $\phi > 4$), to ensure the method 500 does not converge prematurely.

[00065] A suitable selection of the inertia constant w provides a balance between global and local explorations of the objective function. In one implementation of the velocity update rule embodied in equation (9), the inertia constant w is dynamically set to decrease over time as follows:

$$w = w_{\max} - \left(w_{\max} - w_{\min}\right)\frac{t}{T} \quad (7)$$

- where T is the maximum number of iterations.

[00066] In standard PSO algorithms, the position update rule is given by

$$x_j^p(t) = x_j^p(t-1) + v_j^p(t) \quad (8)$$

[00067] However, such a position update rule is not suitable for binary-valued particle positions $\mathbf{x}^p(t)$, as in the method 500.

[00068] In one implementation, the PSO-based algorithm used by the training process 140 is a spiral PSO with wavelet mutation (SPSOWM) algorithm. According to SPSOWM, the velocity update rule used at step 550 of the method 500 is a variant of equation (5) given by:

$$v_j^p(t) = k\left(wv_j^p(t-1) + \phi_1 r_1\left(\bar{x}_j^{p+t \bmod P} - x_j^p(t-1)\right) + \phi_2 r_2\left(\bar{x}_j - x_j^p(t-1)\right)\right) \quad (9)$$

[00069] The difference between equation (9) and the velocity update rule of equation (5) is that the "target" best position for the particle indexed by p is not fixed as the best position $\bar{\mathbf{x}}^p$ for that particle, as in equation (5), but varies over the iterations t to the best positions of other particles in the swarm, i.e. $\bar{\mathbf{x}}^{(p+t) \bmod P}(t)$, where P is the number of particles in the swarms. (The modulo-P operation ensures that the index of the "target" best position for the particle indexed by p, which changes with every iteration t, is always in the range [0, P-1].)

[00070] In conventional PSO algorithms, the velocity and position update rules embodied in equations (5) and (8) cause each particle to tend to move directly towards its own best position $\bar{\mathbf{x}}^p$ and the global best position $\bar{\mathbf{x}}$, often leading to stagnation as all particles catch up with the global best position $\bar{\mathbf{x}}$ and stop moving before the true optimal value is reached.

[00071] In SPSOWM, by contrast, the velocity update rule used at step 550 of the method 500 is defined by equation (9) such that each particle tends to follow a spiral path towards the best position of a *different* particle, as well as the global best position $\bar{\mathbf{x}}$.

[00072] Figs. 6A and 6B illustrate the difference between typical particle paths in conventional PSO (Fig. 6A) and SPSOWM (Fig. 6B), as particles (e.g. 610 and 660) tend to move towards the global best position (600 and 650) in straight-line and spiral paths (620 and 670) respectively.

[00073] According to SPSOWM, the position update rule used at step 555 of the method 500 is defined as

$$x_j^p(t) = \begin{cases} 1, r < Sigmoid\left(v_j^p(t)\right) \\ 0, \text{otherwise} \end{cases} \quad (10)$$

[00074] where r is a random number between 0 and 1, where r less than a certain value (e.g., 1) is applied, and the sigmoid function is defined as

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{11}$$

[00075] Also, according to the SPSOWM, the position update of step 555 is cancelled if a particle that is not the global best particle reaches the position $\bar{\mathbf{x}}$ of the global best particle as a result of the position update applied at step 555. That is,

$$\mathbf{x}^p(t) \rightarrow \mathbf{x}^p(t-1) \text{if } \left(\mathbf{x}^p(t-1) \neq \bar{\mathbf{x}}\right) \text{and } \left(\mathbf{x}^p(t) = \bar{\mathbf{x}}\right) \tag{12}$$

[00076] Equation (12) prevents a particle whose position $\mathbf{x}^p(t)$ is different from the global best particle position $\bar{\mathbf{x}}$ from ever reaching the global best particle position $\bar{\mathbf{x}}$.

[00077] The mutation rule used at step 560 of the method 500 is applied dependent on a predetermined probability of mutation, p_m, between 0 and 1. For each component $x_j^p(t)$ of the position $\mathbf{x}^p(t)$ of each particle at each iteration t, a random number evenly distributed between 0 and 1 is generated. If the generated random number is less than the predetermined probability of mutation p_m, the mutation rule will be applied to that component $x_j^p(t)$.

[00078] According to SPSOWM, the mutation rule used at step 560 of the method 500 is a wavelet mutation rule. The wavelet mutation rule operates on the position $x_j^p(t)$ of the j-th component of the p-th particle as follows: a random number evenly distributed between 0 and 1 is generated. If the generated random number is less than a predetermined function of $x_j^p(t)$, the component $x_j^p(t)$ is set to 1; otherwise, the component $x_j^p(t)$ is set to 0. The predetermined function is $Sigmoid\left(wm_j^p(t)\right)$, where the sigmoid function is defined as in equation (11).

[00079] and its argument $wm_j^p(t)$ is defined as

$$wm_j^p(t) = \begin{cases} \sigma\left(1 - x_j^p(t)\right), \sigma > 0 \\ \sigma x_j^p(t), \sigma \leq 0 \end{cases} \tag{13}$$

[00080] where σ is a mutation parameter. If σ is positive approaching 1, the mutated position $x_j^p(t)$ of the particle will tend to the maximum value of 1. Conversely, when σ is negative approaching -1, the mutated position $x_j^p(t)$ of the particle will tend to the minimum value of 0.

A larger value of $|\sigma|$ gives a larger searching space for the particle position $x_j^p(t)$. When $|\sigma|$ is small, it gives a smaller searching space for fine-tuning the particle position $x_j^p(t)$.

[00081] The mutation parameter σ is defined by

$$\sigma = \frac{1}{\sqrt{a}}\psi\left(\frac{\varphi}{a}\right) \tag{14}$$

[00082] where φ is a random number evenly distributed between -2.5a and 2.5a, ψ is the Morelet wavelet function defined as

$$\psi(x) = e^{-\frac{x^2}{2}}\cos(5x) \tag{15}$$

[00083] and a is a dilation parameter. The value of the dilation parameter a is set to vary with the value of t/T to meet the fine-tuning purpose, where T is the total number of iterations. To perform a local search when t is large, the value of a should increase as t/T increases to reduce the significance of the mutation. Hence, a is defined by a monotonic increasing function of t/T defined as follows:

$$a = \exp\left(-\ln(g)\left(1-\frac{t}{T}\right)^{\zeta_{wm}} + \ln(g)\right) \tag{16}$$

[00084] where ζ_{wm} is the shape parameter of the monotonic increasing function, and g is the upper limit of the dilation parameter a.

[00085] Because of the symmetry property of the Morlet wavelet in equation (15), the overall positive mutation and the overall negative mutation throughout the evolution are nearly the same. This property gives better solution stability (a smaller standard deviation of the solution values upon many trials).

CLAIMS

1. An intrusion detection system comprising:

a flow collector configured to collect packets destined for a computing device and produce an input vector for each detected flow of packets;

a block-based neural network structure developed and configured to label each input vector as either normal or an intrusion,

a FPGA-based hardware system in which said block-based neural network structure is embedded to present a flexible structure for identifications of time-varying nature of intruders and new and unknown intrusions;

a database configured to store labelled input vectors;

a computing device configured to execute a real-time detection process, wherein a pre-training process is configured to process the labelled input vectors to generate a configuration for the block-based neural network structure; and

a detection device that operates with stable convergence rates under a flexible structure in which a high detection rate is maintained with low false alarm rate.

2. The system as claimed in claim 1, wherein the high detection rate reaches around 99%, and the low false positive rate (false alarm rate) is lower than 5%.

3. A method of generating a configuration for a block-based neural network structure, the method comprising:

a. initializing a position and a velocity of each component of each particle in a particle swarm, the weighted structure encoded in a 2-D array representation as the configuration of a block-based neural network structure;

b. developing new movements of particles to solve a stagnation problem;

c. determining whether a termination condition is met by the particle swarm, and if so, returning the configuration represented by a global best position of all the particles in the swarm; otherwise

d. updating the velocity of each component of each particle using a velocity update rule;

e. updating the position of each component of each particle using a position update rule and the corresponding updated component velocity;

f. mutating at least one particle using a mutation rule;

g. reproducing the swarm using the mutated particles;

h. updating the global best position of all the particles in the reproduced swarm; and

i. repeating steps a. to g. until the termination condition is met;

wherein the velocity update rule uses as a target best position for the particle a best position for a particle other than the current particle being updated.

4. The method as claimed in claim 3, wherein in respect of stagnation problem, "stagnation" means immaturely converged.

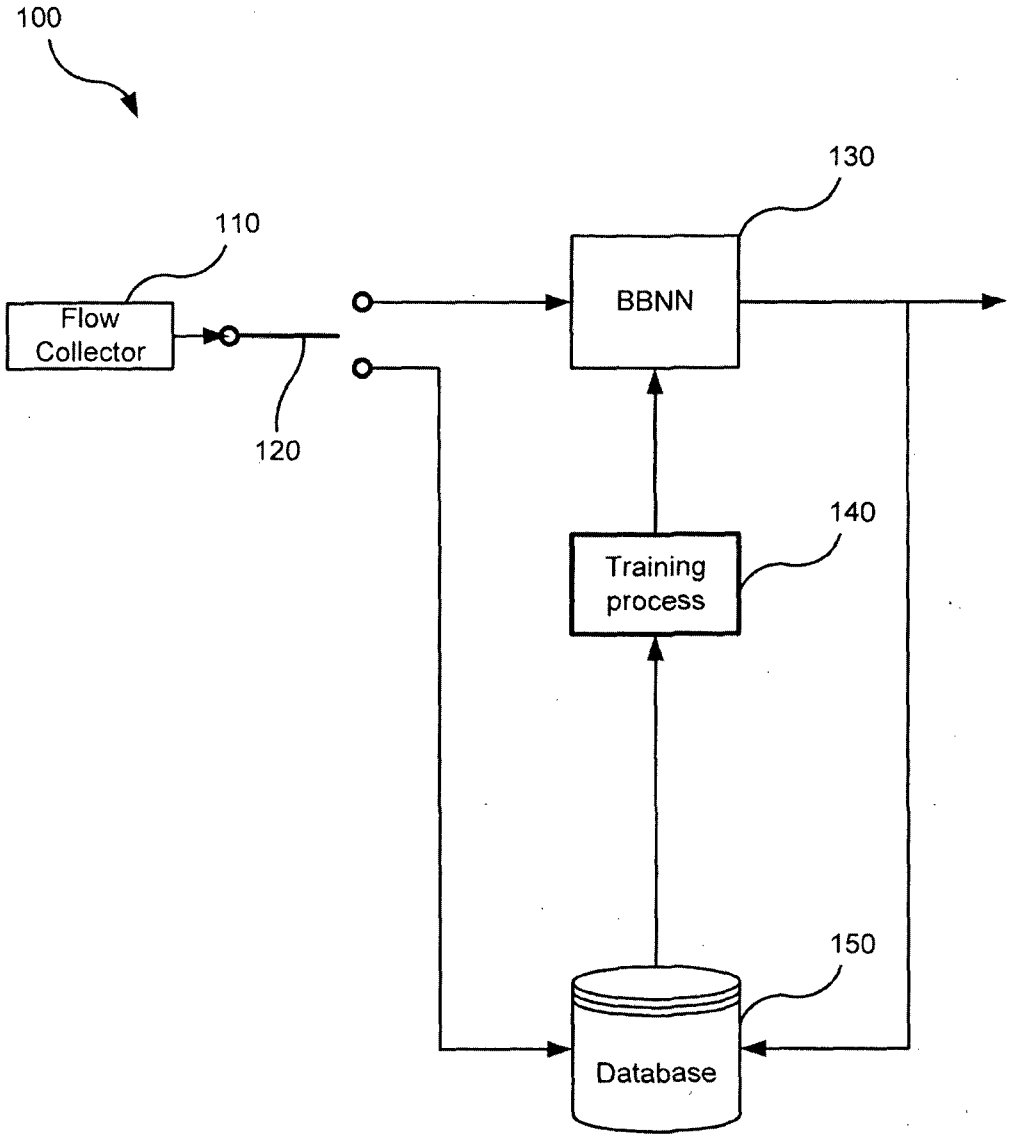
100
130
110
BBNN
Flow
Collector
120
140
Training
process
150
Database

Fig. 1

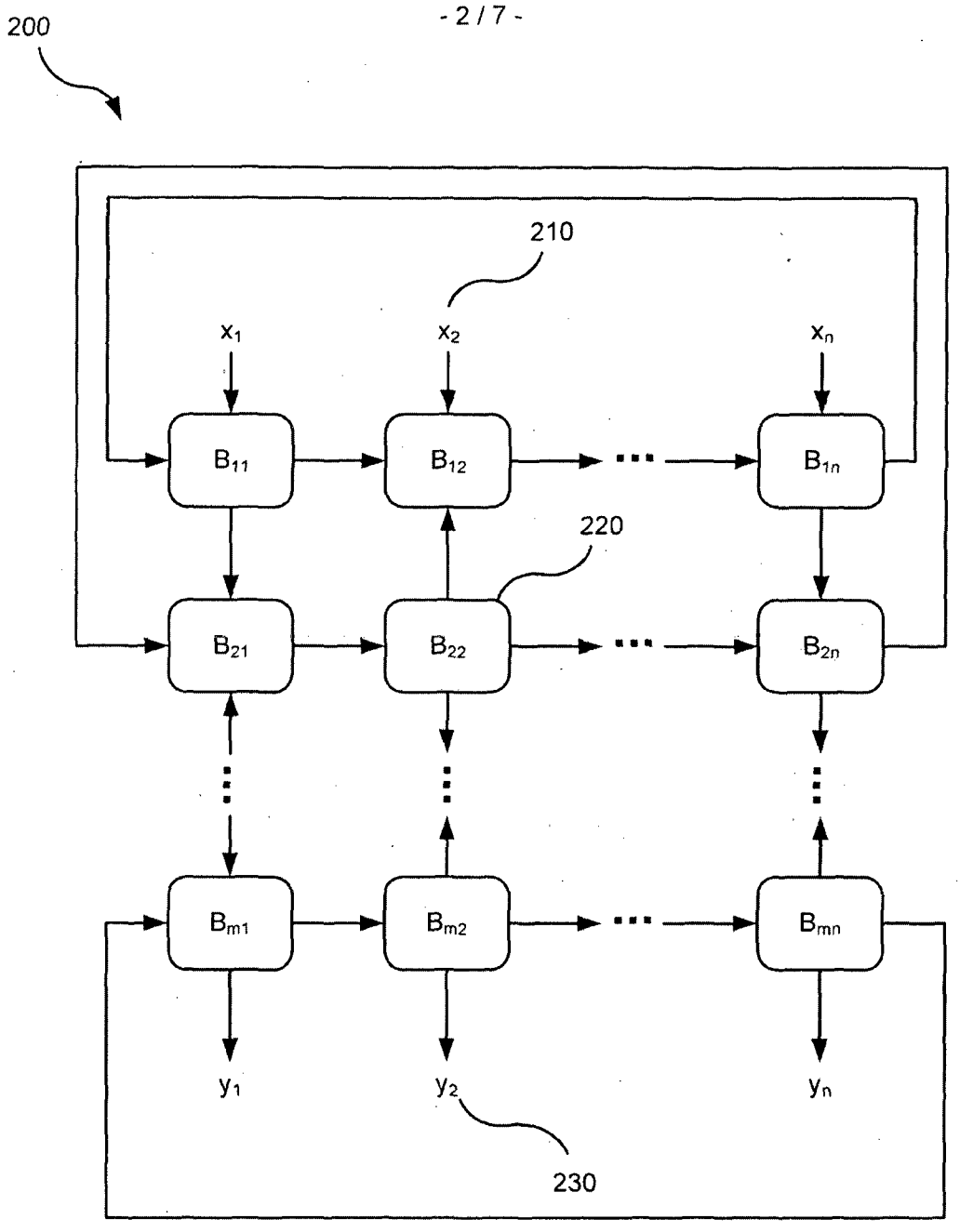
200
210
x_1
x_2
x_n
B_{11}
B_{12}
B_{1n}
220
B_{21}
B_{22}
B_{2n}
B_{m1}
B_{m2}
B_{mn}
y_1
y_2
y_n
230

Fig. 2

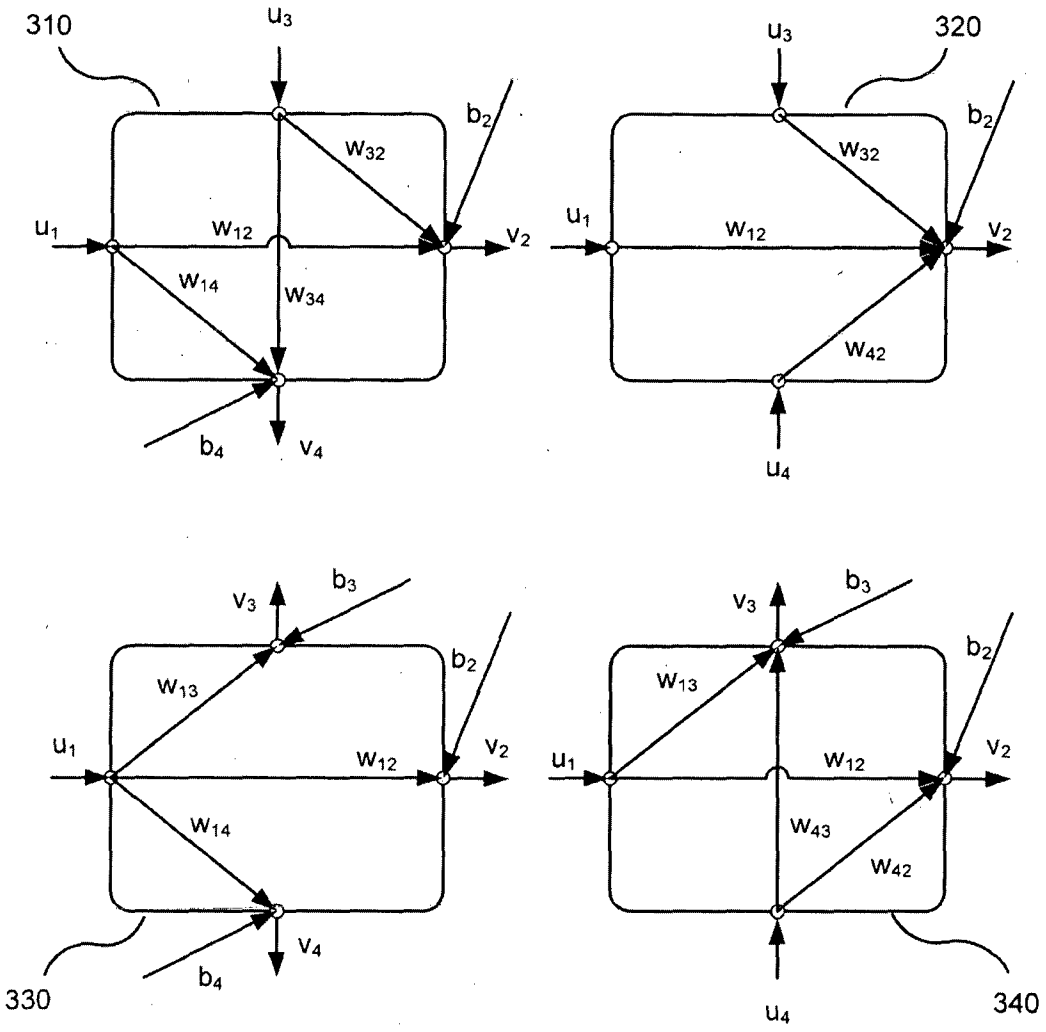
310
320
330
340
u1
u3
u4
v2
v3
v4
b2
b3
b4
w12
w13
w14
w32
w34
w42
w43

Fig. 3

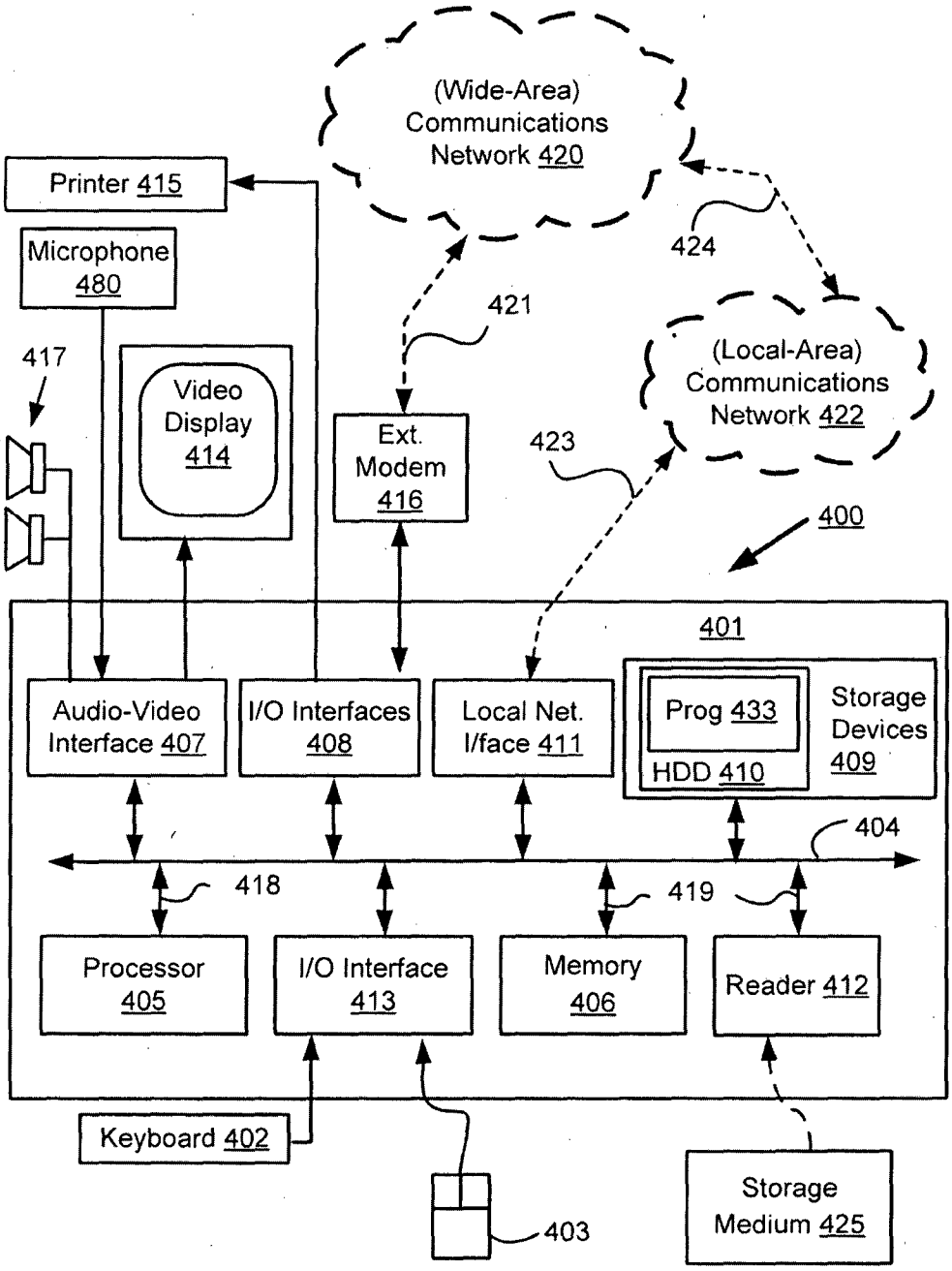
(Wide-Area)
Communications
Network 420
Printer 415
Microphone
480
424
421
417
Video
Display
414
Ext.
Modem
416
(Local-Area)
Communications
Network 422
423
400
401
Audio-Video
Interface 407
I/O Interfaces
408
Local Net.
I/face 411
Prog 433
HDD 410
Storage
Devices
409
404
418
419
Processor
405
I/O Interface
413
Memory
406
Reader 412
Keyboard 402
403
Storage
Medium 425

Fig. 4A

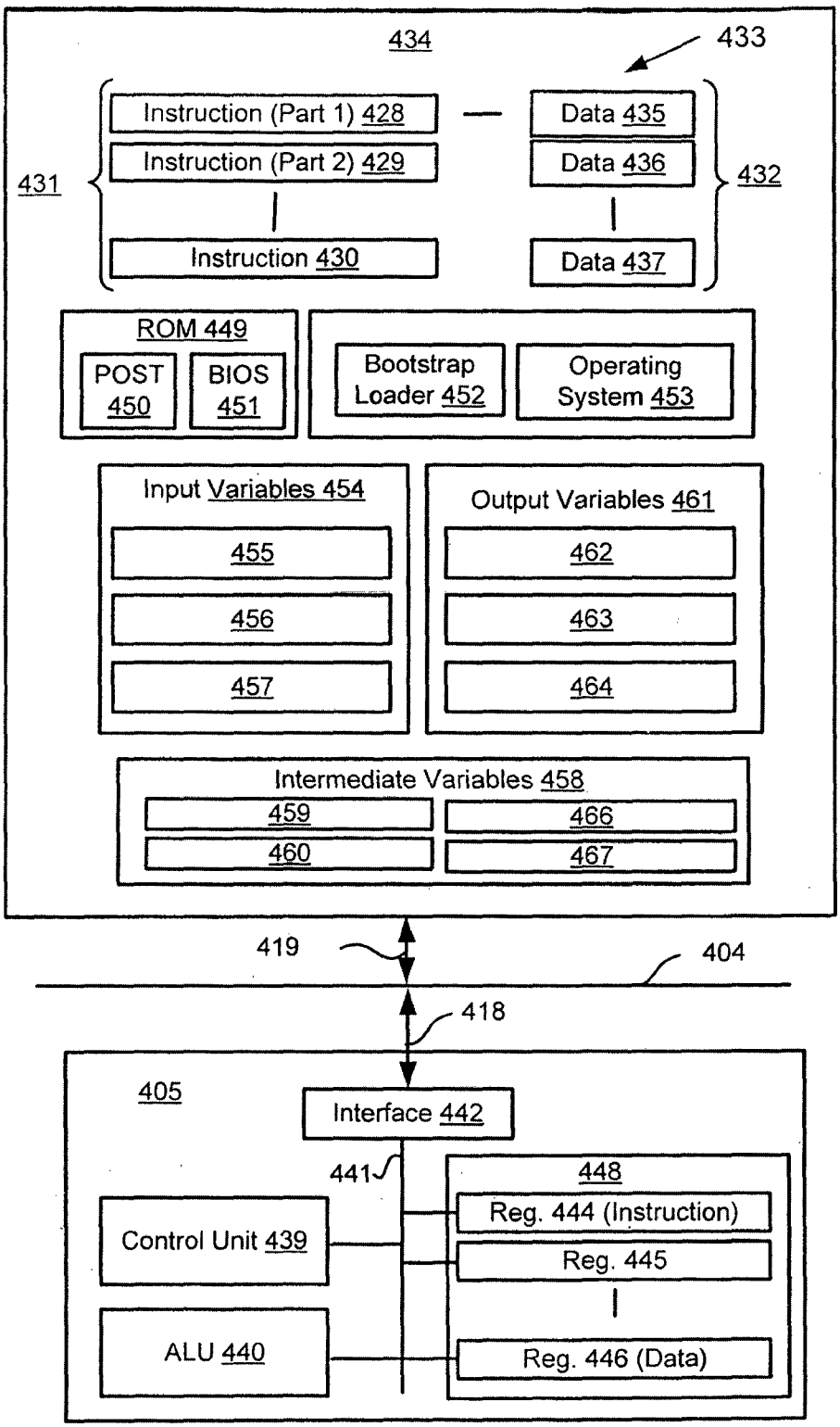
434
433
Instruction (Part 1) 428
Instruction (Part 2) 429
Instruction 430
Data 435
Data 436
Data 437
431
432
ROM 449
POST 450
BIOS 451
Bootstrap Loader 452
Operating System 453
Input Variables 454
455
456
457
Output Variables 461
462
463
464
Intermediate Variables 458
459
460
466
467
419
404
418
405
Interface 442
441
448
Reg. 444 (Instruction)
Reg. 445
Reg. 446 (Data)
Control Unit 439
ALU 440

Fig. 4B

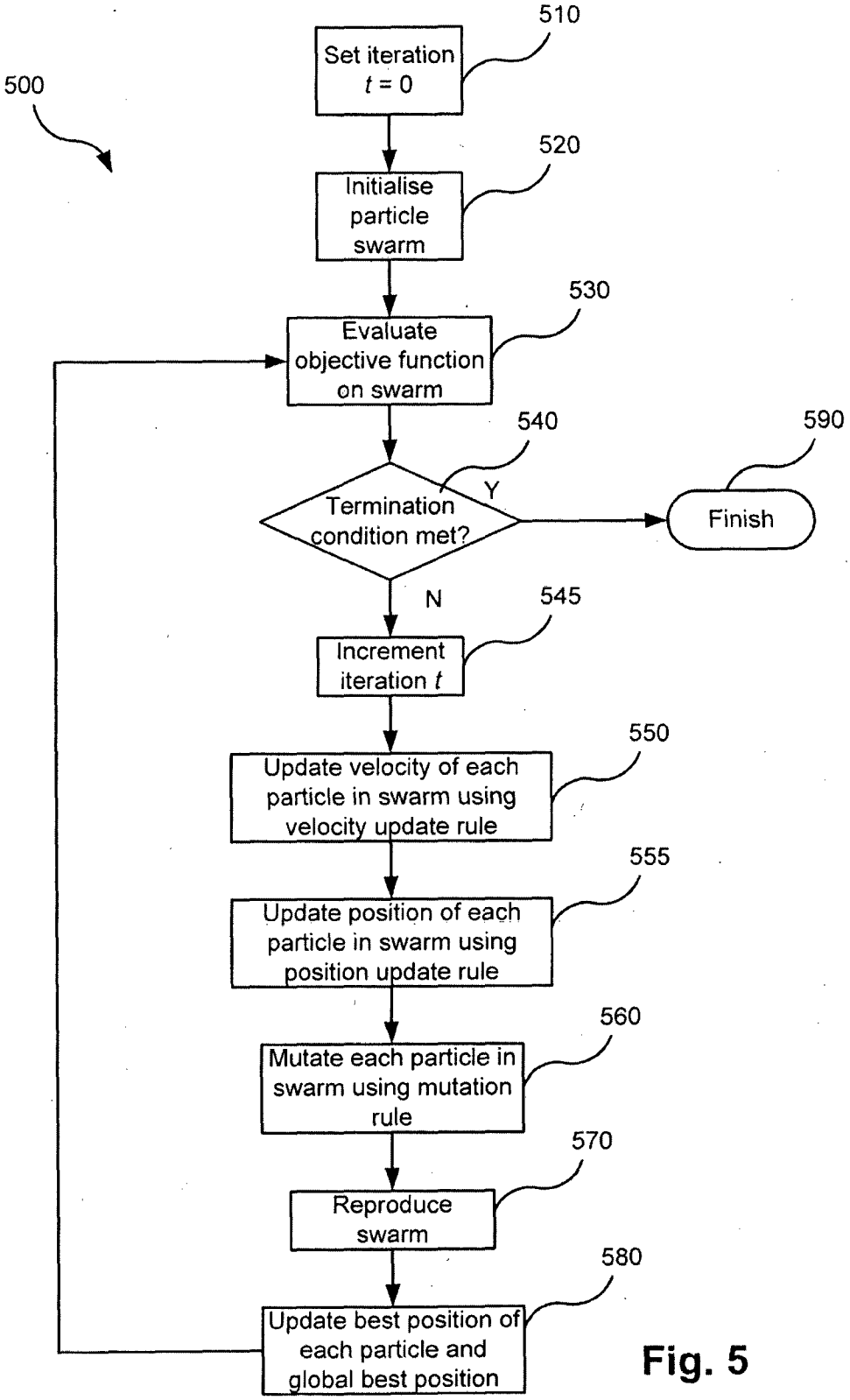
500
Set iteration $t = 0$
510
Initialise particle swarm
520
Evaluate objective function on swarm
530
Termination condition met?
540
Y
Finish
590
N
Increment iteration t
545
Update velocity of each particle in swarm using velocity update rule
550
Update position of each particle in swarm using position update rule
555
Mutate each particle in swarm using mutation rule
560
Reproduce swarm
570
Update best position of each particle and global best position
580

Fig. 5

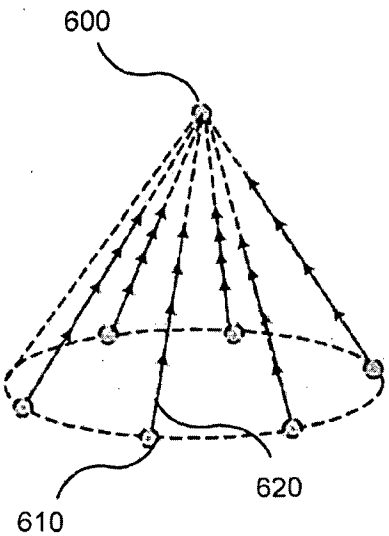

Fig. 6A

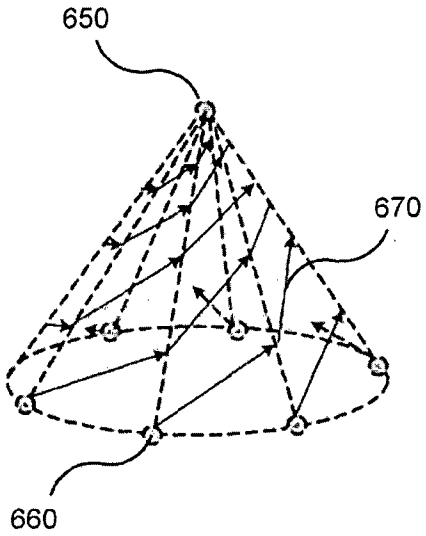

Fig. 6B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU2013/001430

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/76 (2013.01)

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)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, EPODOC, Espacenet, Google, Google Patents, with keywords (intrusion, attack, detection, prevention, block-based neural network, FPGA) and the like terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	

[X] Further documents are listed in the continuation of Box C [X] 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
25 February 2014	25 February 2014

Name and mailing address of the ISA/AU	**Authorised officer**
AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA Email address: pct@ipaustralia.gov.au Facsimile No.: +61 2 6283 7999	Ying Li AUSTRALIAN PATENT OFFICE (ISO 9001 Quality Certified Service) Telephone No. 0262832802

INTERNATIONAL SEARCH REPORT C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		International application No. **PCT/AU2013/001430**
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	TRAN, Q A et al, A Real-time NetFlow-based Instrusion Detection System with Improved BBNN and High-Frequency Field Programable Gate Arrays, IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, pp.201-208. June 2012. abstract, section II(A), section III(A), fig. 4, p. 202 par. 1	1-2

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU2013/001430

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:
the subject matter listed in Rule 39 on which, under Article 17(2)(a)(i), an international search is not required to be carried out, including

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a)

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

See Supplemental Box for Details

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:
1-2

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT	International application No. PCT/AU2013/001430

Supplemental Box

Continuation of: **Box III**

This International Application does not comply with the requirements of unity of invention because it does not relate to one invention or to a group of inventions so linked as to form a single general inventive concept.

This Authority has found that there are different inventions based on the following features that separate the claims into distinct groups:

- Claims 1-2 are directed to an intrusion detection system with a high detection rate. The features of *collecting packets destined for a computing device, producing an input vector for each detected flow of packets, labelling each input vector, presenting a flexible structure for identifications of time-varying nature of intruders and new and unknown intrusions, processing the labelled input vectors, and operating with stable convergence rates* are specific to this group of claims.

- Claims 3-4 are directed to a method of generating a configuration for a block-based neural network structure. The features of *initializing a position and a velocity of each component of each particle in a particle swarm, the weighted structure encoded in a 2-D array representation, developing new movements of particles, determining whether a termination condition is met by the particle swarm, returning the configuration represented by a global best position of all the particles in the swarm, updating the velocity of each component of each particle using a velocity update rule, updating the position of each component of each particle using a position update rule and the corresponding updated component velocity, mutating at least one particle using a mutation rule, reproducing the swarm using the mutated particles, and updating the global best position of all the particles in the reproduced swarm* are specific to this group of claims.

PCT Rule 13.2, first sentence, states that unity of invention is only fulfilled when there is a technical relationship among the claimed inventions involving one or more of the same or corresponding special technical features. PCT Rule 13.2, second sentence, defines a special technical feature as a feature which makes a contribution over the prior art.

When there is no special technical feature common to all the claimed inventions there is no unity of invention.

In the above groups of claims, the identified features may have the potential to make a contribution over the prior art but are not common to all the claimed inventions and therefore cannot provide the required technical relationship. The only feature common to all of the claimed inventions is block-based neural network structure. However it is considered that this feature is generic in this particular art.

Therefore this common feature cannot be a special technical feature. Hence there is no special technical feature common to all the claimed inventions and the requirements for unity of invention are consequently not satisfied *a priori*.

INTERNATIONAL SEARCH REPORT Information on patent family members	International application No. **PCT/AU2013/001430**

This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document/s Cited in Search Report		Patent Family Member/s	
Publication Number	**Publication Date**	**Publication Number**	**Publication Date**

End of Annex

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.