CLASS-BASED DISTRIBUTED EVOLUTIONARY ALGORITHM FOR ASSET MANAGEMENT AND TRADING

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] The present application claims benefit under 35 USC 119(e) of U.S. provisional application number 61/173581, filed April 28, 2009, entitled "Distributed Evolutionary Algorithm for Stock Trading", and U.S. provisional application number 61/173582, filed April 28, 2009, entitled "Distributed Evolutionary Algorithm for Stock Trading" the contents of both of which are incorporated herein by reference in their entirety.

BACKGROUND OF THE INVENTION

[0002] Complex financial trend and pattern analysis processing is conventionally done by supercomputers, mainframes or powerful workstations and PCs, typically located within a firm's firewall and owned and operated by the firm's Information Technology (IT) group. The investment in this hardware and in the software to run it is significant. So is the cost of maintaining (repairs, fixes, patches) and operating (electricity, securing data centers) this infrastructure.

[0003] Stock price movements are generally unpredictable but occasionally exhibit predictable patterns. Genetic Algorithms (GA) are known to have been used in stock categorization. According to one theory, at any given time, 5% of stocks follow a trend. Genetic algorithms are thus sometimes used, with some success, to categorize a stock as following or not following a trend.

[0004] Evolutionary algorithms, which are supersets of Genetic Algorithms, are good at traversing chaotic search spaces. As has been shown by Koza, J.R., "Genetic Programming: On the Programming of Computers by Means of Natural Selection", 1992, MIT Press, an evolutionary algorithm can be used to evolve complete programs in declarative notation. The basic elements of an evolutionary algorithm are an environment, a model for a gene, a fitness function, and a reproduction function. An environment may be a model of any problem statement. A gene may be defined by a set of rules governing its behavior within the environment. A rule is a list of conditions followed by an action to be performed in the environment. A fitness function may be defined by the degree to which an evolving rule set is

successfully negotiating the environment. A fitness function is thus used for evaluating the fitness of each gene in the environment. A reproduction function generates new genes by mixing rules with the fittest of the parent genes. In each generation, a new population of genes is created.

[0005] At the start of the evolutionary process, genes constituting the initial population are created entirely randomly, by putting together the building blocks, or alphabets, that form a gene. In genetic programming, the alphabets are a set of conditions and actions making up rules governing the behavior of the gene within the environment. Once a population is established, it is evaluated using the fitness function. Genes with the highest fitness are then used to create the next generation in a process called reproduction. Through reproduction, rules of parent genes are mixed, and sometimes mutated (i.e., a random change is made in a rule) to create a new rule set. This new rule set is then assigned to a child gene that will be a member of the new generation. In some incarnations, the fittest members of the previous generation, called elitists, are also copied over to the next generation.

BRIEF SUMMARY OF THE INVENTION

[0006] A networked computer system, in accordance with one embodiment of the present invention, includes one or more sever computers and a multitude of client computers each of which is assigned to a different class with each class being defined by a subset of indictors. Each client computer includes, in part, a memory, a communication port, and a processor. The memory in each client computer is operative to store a multitude of genes each characterized by a set of conditions and the subset of indicators associated with the class to which the client computer is assigned. The communication port in each client computer continues to periodically receive data associated with the genes stored in the memory. The processor in each client computer evaluates the performance characteristic of each of its genes by comparing a solution provided by that gene with the periodically received data associated with that gene. Accordingly, the performance characteristic of each gene is updated and varied with each periodically received data. The performance characteristic of a gene defines its fitness.

[0007] In one embodiment, the data associated with each gene is historical trading data and the solution provided by each gene is a trade recommended by the gene. In one embodiment at least two of the subsets of indicators are overlapping indicators. In one embodiment, genes

whose fitness are determined as falling below a first predefined threshold value following an evaluation covering a first time period are discarded. The remaining (surviving) genes continue to be evaluated by their client computers as new data is received on a periodic basis.

[0008] In one embodiment, genes that survive the first evaluation time period continue to be evaluated by the client computers for one or more additional time periods in response to instructions from the server computer. During each additional evaluation period, genes whose fitness fall below a threshold value are discarded. Genes that survive the one or more evaluation periods, as requested by the server, are stored in an elitist gene pool for selection by the server. The threshold values used to evaluate a gene's fitness corresponding to multiple time periods may or may not be equal.

[0009] In one embodiment, the server computer selects genes from the clients computers' elitist pool and stores them in its memory. The server may send the genes it receives from any class of clients back to the clients of the same class for further evaluation covering additional time periods. Such client computers perform further evaluation of the genes for the additional time periods and attempt to send the surviving genes back to the server. Genes that are discarded by the client computers are reported to the server. In one embodiment, the server only receives genes whose fitness as determined by the client computers are equal to or greater than the fitness of the genes previously stored by the server.

[0010] In one embodiment, genes initially evaluated by the client computers are generated in accordance with computer instructions stored and executed by the client computers. In one embodiment, the server stores a fixed number of genes in its memory at any given time. The server, after accepting a new gene from a client computer, combines the fitness value of the accepted gene with a corresponding fitness value the server has previously stored in the server for that gene.

[0011] A method of solving a computational problem, in accordance with one embodiment of the present invention, includes in part, storing a multitude of genes each characterized by a number of conditions, a subset of indicators and a gene class; continuing to periodically receive data associated with the genes; and evaluating performance characteristic of each gene by comparing a solution provided by the gene with the periodically received data associated with that gene. Accordingly, the performance characteristic of each gene is updated and varied with each periodically received data. The performance characteristic of a gene defines its fitness.

[0012] In one embodiment, the data associated with each gene is historical trading data and the solution provided by each gene is a trade recommended by the gene. In one embodiment at least two of the subsets of indicators are overlapping indicators. In one embodiment, genes whose fitness are determined as falling below a first predefined threshold value following an evaluation covering a first time period are discarded. The remaining (surviving) genes continue to be evaluated by their client computers as new data is received on a periodic basis.

[0013] In one embodiment, genes that survive the first evaluation time period continue to be evaluated for one or more additional time periods in response to instructions. During each additional evaluation period, genes whose fitness fall below a threshold value are discarded. Genes that survive the one or more evaluation periods are stored in an elitist gene pool for selection. The threshold values used to evaluate a gene's fitness corresponding to multiple time periods may or may not be equal.

[0014] In one embodiment, the server computer selects genes from the clients computers' elitist pool and stores them in its memory. The server may send the genes it receives from any class of clients back to the clients of the same class for further evaluation covering additional time periods. Such client computers perform further evaluation of the genes for the additional time periods and attempt to send the surviving genes back to the server. Genes that are discarded by the client computers are reported to the server. In one embodiment, the server only receives genes whose fitness as determined by the client computers are equal to or greater than the fitness of the genes previously stored by the server.

[0015] In one embodiment, genes initially evaluated by the client computers are generated in accordance with computer instructions stored and executed by the client computers. In one embodiment, the server stores a fixed number of genes in its memory at any given time. The server, after accepting a new gene from a client computer, combines the fitness value of the accepted gene with a corresponding fitness value the server has previously stored in the server for that gene.

[0016] In one embodiment, genes whose fitness are determined as falling below a first predefined threshold value following an evaluation covering a first time period spanning P days are discarded. The remaining genes that survive the evaluation continue to be evaluated as new data are received on a periodic basis.

[0017] In one embodiment, genes that survive the first evaluation time period continue to be evaluated for one or more additional time periods in response to instructions. During each

additional evaluation period, genes whose fitness fall below a threshold value are discarded. Genes that survive the one or more evaluation periods are stored in an elitist gene pool for possible selection. The selected genes are stored in a memory by a server computer. The threshold values used to evaluate a gene's fitness corresponding to multiple time periods may or may not be equal.

[0018] In one embodiment, the selected genes that are stored by the server computer are sent back for further evaluation--covering additional time periods—only to client computers that previously evaluated these genes and thus have the same class as the genes they receive. Genes that survive this further evaluation are provided for selection. Genes that do not survive the further evaluation are discarded but noted in a report. In one embodiment, only genes whose fitness is determined as being equal to or greater than the fitness of previously stored genes are selected for storage. In another embodiment, the selected genes that are stored by the server computer may be sent back for further evaluation to client computers that did not previously evaluate these genes as long as the client class is the same as the gene class.

[0019] In one embodiment, the genes are generated in accordance with computer instructions stored and executed by the client computers. In one embodiment, a fixed number of selected genes are stored at any given time by the server computer. In one embodiment, the fitness value of a newly selected gene by the server computer is combined with a corresponding fitness value of the same gene if that gene was previously selected and stored by the server computer.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] Figure 1 is an exemplary high-level block diagram of a network computing system configured to execute an evolutionary algorithm, in accordance with one embodiment of the present invention.

[0021] Figure 2 shows a number of functional logic blocks of the client and server computer system of Figure 1, in accordance with one exemplary embodiment of the present invention.

[0022] Figure 3 shows an exemplary convergence factor as a function of evaluation time for a given pool.

[0023] Figure 4 shows a network computer system having a sever and a multitude of clients forming a multitude of different classes, in accordance with one exemplary embodiment of the present inventions.

[0024] Figures 5A shows an exemplary flowchart for evaluating performance characteristics of a number of genes by one or more client computers, in accordance with one embodiment of the present invention.

[0025] Figures 5B shows an exemplary flowchart for evaluating performance characteristics of a number of genes by one or more server computers, in accordance with one embodiment of the present invention.

[0026] Figure 6 shows a number of components of the client and sever computers of Figure 1.

DETAILED DESCRIPTION OF THE INVENTION

[0027] In accordance with one embodiment of the present invention, a server computer and a multitude of client computers form a network computing system that is scalable and is adapted to continue to evaluate the performance characteristics of a number of genes generated using a software application running on the client computers. Each client computer is assigned to one of a multitude of classes. Each class is associated with and represented by a subset of indicators that are used by the client members of that class to create new genes. In some embodiments, two or more classes are merged to generate a new class of genes represented by at least the union of the indicator subsets of the merged classes. Accordingly, in some embodiments the new class is represented by the union of the indicator subsets of the merged classes. In other embodiments, the new class is represented by the union of the indicator subsets of the merged classes in combination with a new subset of indicators different from the indicator subsets of the merged classes. In yet other embodiments, a merge of two classes additionally results in the addition of a new class represented by a different subset of indicators that may be randomly generated. In the following, a server computer is understood to refer to any data processing device having one or more CPUs or GPUs that coordinates, supervises, collects data from, controls, or directs the actions of one or more client computer. For example, in a device having a CPU with four processing core, one processing core may be assigned to be the server while the remaining three processing cores may be assigned as client computers. Alternatively, for example, client computers may be

personal computation/ communication devices that are supervised, are in part controlled by, receive instructions from, etc. from another computing device designated as the server computer. In one embodiment, the genes are virtual asset traders that recommend trading options.

[0028] In the following description it is understood that (i) a system refers to a hardware system, a software system, or a combined hardware/software system; (ii) a network computing system refers to a multitude of mobile or stationary computer systems that are in communication with one another either wirelessly or using wired lines; a network computing system includes, in part, a multitude of computers at least one of which is a central or distributed server computer, with the remaining computers being client computers; each server or client computer includes at least one CPU and a memory.

[0029] Figure 1 is an exemplary high-level block diagram of a network computing system 100, in accordance with one embodiment of the present invention. Network computing system 100 is shown as including, in part, N client computers 20 and one server computer 10. It is understood that server 10 may be a central or a distributed server. A client computer may be a laptop computer, a desktop computer, a cellular/VoIP handheld communication/computation device, a table computer, or the like.

[0030] A broadband connection connects the client computers (alternatively referred to herein as client) 20 to sever computer (alternatively referred to herein as server) 10. Such connection may be cable, DSL, WiFi, 3G wireless, 4G wireless or any other existing or future wireline or wireless standard that is developed to connect a CPU to the Internet. Any CPU may be used if a client software, in accordance with the present invention and as described further below, is enabled to run on that CPU.

[0031] In one embodiment, network computing system 100 implements financial algorithms/analysis and computes trading policies. To achieve this, the computational task associated with the algorithms/analysis is divided into a multitude of sub-tasks each of which is assigned and delegated to a different one of the clients. The computation results achieved by the clients are thereafter collected and combined by server 10 to arrive at a solution for the task at hand. The sub-task received by each client may include an associated algorithm or computational code, data to be implemented by the algorithm, and one or more problems/questions to be solved using the associated algorithm and data. Accordingly, in some examples, server 10 receives and combines the partial solutions supplied by the CPU(s)

disposed in the clients to generate a solution for the requested computational problem. When the computational task being processed by network computing system 100 involves financial algorithms, the final result achieved by integration of the partial solutions supplied by the clients may involve a recommendation on trading of one or more assets. In other examples, the tasks performed by the clients are independent from one another. Accordingly, in such embodiments, the results achieved by the clients are not combined with one another, although the sever pools the results it receives from clients to advance the solution. Although the following description is provided with reference to making recommendations for trading of financial assets (e.g., stocks, indices, currencies, etc.) using genes, it is understood that the embodiments of the present invention are equally applicable to finding solutions to any other computational problem, as described further below.

[0032] Scaling of the evolutionary algorithm may be done in two dimensions, namely by the pool size and/or evaluation. In an evolutionary algorithm, the larger the pool or population of the genes, the greater is the diversity of the genes. Consequently, the likelihood of finding fitter genes increases with increases in pool size. In order to achieve this, the gene pool may be distributed over many clients. Each client evaluates its pool of genes and sends the fittest genes to the server, as described further below.

[0033] Each client that is connected to the network, in accordance with the present invention, receives or downloads a client software. The client software automatically generates a multitude of genes whose number may vary depending on the memory size and the CPU processing power of the client. For example, in one embodiment, a client may have 1000 genes for evaluation.

[0034] A gene is assumed to be a virtual trader that is given a hypothetical sum of money to trade using historical data. Such trades are performed in accordance with a set of rules of that define the gene thereby prompting it to buy, sell, hold its position, or exit its position. A rule is a list of conditions followed by an action, which may be, for example, buy, sell, exit or hold. Rules may also be designed to contain gain-goal and stop-loss targets, thus rendering the exit action redundant. A hold occurs when no rule in the gene is triggered, therefore, the gene effectively holds its current position. A condition is a conjunction list of indicator based conditions. Indicators are the system inputs that can be fed to a condition, such as tick, or the closing price. Indicators could also be introspective to indicate the fitness or other attributes of the gene at any given moment.

[0035] The following code defines a rule within a gene in terms of conditions and indicators, as well as the action taken by the gene, in accordance with one exemplary embodiment of the present invention:

if (PositionProfit >= 2% *and* !(tick= (-54/10000)% prev tick *and* MACD is negative)
and !(tick= (-119/10000)% prev tick *and* Position is long))
and !(ADX x 100 <= 5052))
then SELL

where "and" represents logical "AND" operation, "!" represents logical "NOT" operation, "tick", "MACD" and "ADX" are examples of stock indicators, "SELL" represents action to sell, and "PositionProfit" represents the profit position of the gene.

[0036] Genes are evaluated over stock-days. A stock-day is a day's worth of historical data for a specific stock. At a specific interval in a given stock-day, for example, every 5 minutes, rules of a gene are evaluated by assigning the current values of the indicators into the conditions of each rule. If none of the conditions of a gene are true for the indicator values, the gene holds its previous position. If the gene had no position, it performs no action. A gene may be designed to take the action of its first rule whose conditions are satisfied. If, for example, the rule's action is a sell, then the trade proposed by the gene is taken to be a sell. In another example, a rule that fires with the exit action may trump all other votes and force an exit from the gene's current position.

[0037] In accordance with one embodiment of the present invention, a gene's fitness or success is determined by approximation and using a large amount of data. The model used to evaluate the genes may thus be partial and cover shorter time spans, while improving in accuracy as the genes are evaluated over more stock-days and gain experience. To establish an initial approximation for the genes' fitness, as described further below, the genes' fitness are first evaluated over a subset of the available data. The time period over which a gene's fitness has been evaluated is referred to herein as the gene's maturity age, also referred to herein as the gene's age. Genes that reach a predefined age are enabled to reproduce and contribute to the next generation of genes. Each such genes can continue to live and stay in the gene pool as long as its cumulative fitness meets predefined conditions.

[0038] The historical data used to evaluate a gene's fitness is significant. Therefore, even with today's high processing power and large memory capacity computers, achieving quality

results within a reasonable time is often not feasible on a single machine. A large gene pool also requires a large memory and high processing power. In accordance with one embodiment of the present invention, scaling is used to achieve high quality evaluation results within a reasonable time period. The scaling operation is carried out in two dimensions, namely in pool size as well as in evaluation of the same gene to generate a more diverse gene pool so as to increase the probability of finding fitter genes. Therefore, in accordance with one embodiment of the present invention, the gene pool is distributed over a multitude of clients for evaluation. Each client continues to evaluate its gene pool using historical data that the client periodically receives on a sustained and continuing basis. In other words, a gene's performance (also referred to herein as the genes' fitness) continues to be evaluated over additional historical data that are received periodically and on a continuing basis by the clients. Genes that satisfy one or more predefined conditions are transmitted to the server.

[0039] In accordance with another embodiment of the present invention, gene distribution is also used to increase the speed of evaluation of the same gene. To achieve this, genes that are received by the server but have not yet reached a certain maturity age or have not yet met one or more predefined conditions, may be sent back from the server to a multitude of clients for further evaluation. The evaluation result achieved by the clients (alternatively called herein as partial evaluation) for a gene is transferred back to the server. The server merges the partial evaluation results of a gene with that gene's fitness value at the time it was sent to the clients to arrive at a fitness measure for that gene. For example, assume that a gene is 500 evaluation days old and is sent from the server to, for example, two clients each instructed to evaluate the gene for 100 additional days. Accordingly, each client further evaluates the gene for the additional 100 stock-days and reports its evaluation results to the server. These two results are combined with the gene's fitness measure at the time it was sent to the two clients. The combined results represent the gene's fitness evaluated over 700 days. In other words, the distributed system, in accordance with this example, increases the maturity age of a gene from 500 days to 700 days using only 100 different evaluation days for each client. A distributed system, in accordance with the present invention, is thus highly scalable in evaluating its genes.

[0040] Advantageously, in accordance with the present invention, clients are enabled to use the genes stored in the server in their local reproductions, thereby improving the quality of their genes. Each client is a self-contained evolution device, not only evaluating the genes in

its pool, but also creating a new generation of genes and moving the evolutionary process forward locally. Since the clients continue to advance with their own local evolutionary process, their processing power is not wasted even if they are not in constant communication with the server. Once communication is reestablished with the server, clients can send in their fittest genes to the server and receive genes from the server for further evaluation.

[0041] Each client computer has a communication port to access one or more data feed servers, generally shown using reference numeral 30, to obtain information required to solve the problem at hand. When recommending trading strategies for assets such as stocks, commodities, currencies, and the like, the information supplied by the data feed servers includes the asset values covering a specified time period. Alternatively, although not shown, the information required to solve the problem at hand may be supplied from a data feed server 30 to the clients 20 via server 10. Although server 10 is shown as a single central server in Figure 1, it is understood that server 10 may be a distributed server.

[0042] Figure 2 shows a number of logic blocks of each client 20 and server 10. As is seen, each client 20 is shown as including a pool 24 of genes that are generated by a self-contained application software running on the client. In the following, each gene is assumed to be a trader of financial asset (e.g., stock), although it is understood that a gene may generally be suited to finding solutions to any other computational problem. The performance characteristics of each gene of a client is evaluated over a first predefined a time period, spanning P trading days, e.g. 600 days, using evaluation block 22. The evaluation for each gene is performed by comparing the trading recommendations of that gene and determining its corresponding rate of return over the predefined time period. The performance characteristic of a gene is alternatively referred to herein as the gene's fitness. Client 20 receives historical trading data to determine the fitness of its genes.

[0043] Upon completion of the performance evaluation of all its genes, each client computer selects and places its best performing genes (surviving genes) in elitist pool 26. In one embodiment, the surviving genes may be, e.g., the top 5% performers of the gene pool as determined by the rate of return of their recommendations. In other embodiments, the surviving genes are genes whose fitness exceeds a predefined threshold value. The remaining genes that fail to meet the required conditions for fitness do not survive and are discarded. Each client continues to evaluate its elitist (surviving) genes using the historical trading data that the client continues to receive on a periodic basis.

[0044] In some embodiments, following the initial evaluation of the genes over the first P trading days, the surviving genes are further evaluated for a multitude S of additional time periods each spanning Q other trading days. For example, following the initial evaluation of the genes during the first 600 trading days, each surviving gene is further evaluated over two additional time periods, each spanning 600 more trading days. Therefore, in such examples, each gene is evaluated for 1800 trading days. Such multitude of time periods may be non-overlapping consecutive time periods. Furthermore, the number of trading days, i.e. Q, of each additional time period may or may not be equal to the number of trading days, i.e. P, of the initial evaluation period. Evaluation in each such additional time period may result in discarding of genes that have survived previous evaluations. For example, a gene that may have survived the initial evaluation period of, e.g. 600 days, may not survive the evaluation carried out during the second time period of, e.g. 600 days, if its fitness during the trading days spanning the, e.g. 1200 days, is below a predefined threshold level. Genes stored in the elitist pool 26 that fail to survive such additional evaluation periods are discarded. The fitness threshold level that is required to pass the initial evaluation period may or may not be the same as the fitness threshold levels required to pass successive evaluations.

[0045] Genes that survive the fitness conditions of the initial and successive evaluation periods remain stored in elitist pool 26 and are made available to gene selection block 28 for possible selection and transmission to server 10, as shown in Figure 2. Genes received by server 10 from client computers are stored in sever gene pool 14 of server 10. Gene selection block 28 compares the fitness of the genes in its associated elitist pool 26 with those of the worst performing genes stored in pool 14. In one embodiment, server 10 only accepts genes whose fitness, as determined by a client computer, is at least equal to or better than the fitness of the genes stored in gene pool 14. Server 10 thus informs the client computer about the fitness of its worst performing genes to enable the gene selection module 28 make this comparison and identify genes that server 10 will accept. For example, server 10 may send an inquiry to gene selection module 28 stating "the fitness of my worst gene is X, do you have better performing genes?" Gene selection module 28 may respond by saying "I have these 10 genes that are better" and attempt to send those genes to the server. In one embodiment, gene pool 14 has a fixed size. Therefore in order to accepting a new gene, server 10 discards one of the genes stored in its pool 14. In one embodiment, the initial population of pool 14 is formed from the fittest of all the genes initially stored in the clients' collective elitist pools. This process continues until pool 14 reaches its full capacity that may dynamically vary. In

another embodiment, to form its initial gene population, pool 14 continues to accept genes stored in the elitist pools until pool 14 reaches its full capacity.

[0046] Gene acceptance block 12 is configured to ensure that a gene arriving from a client has a better fitness than the genes already stored in server pool 14 before that gene is added to server pool 358. Gene acceptance block 12 stamps each accepted gene with an ID, and performs a number of house cleaning operations prior to adding the accepted gene to server pool 14.

[0047] Genes in elitist pool 26 are allowed to reproduce. To achieve this, gene reproduction block 30 randomly selects and combines two or more genes, i.e., by mixing the rules used to create the parent genes . Pool 24 is subsequently repopulated with the newly created genes (children genes) as well as the genes that were in the elitist pool. The old gene pool is discarded. The new population of genes in pool 24 continue to be evaluated as described above.

[0048] In some embodiments, server 10 sends each genes stored in pool 14 whose maturity age (i.e., the sum of the trading days over which a gene's fitness has been evaluated) is less than a predefined value back to a group of selected client computers for more fitness evaluation over additional time periods spanning W trading days. Genes whose fitness as evaluated over the additional W trading days fail to satisfy one or more predefined conditions, e.g., their fitness is less than a required a threshold value, are discarded by the client computers. Genes whose fitness as evaluated over the additional W trading days satisfy the one or more predefined conditions are sent back to the server 10 for storage in pool 14. The discarded genes are reported to the server by the client computers.

[0049] In some embodiments, to increase the age if a gene(s) stored in pool 14, server 10 sends the gene to a number of client computers each instructed to perform further evaluation of the gene over a different set of trading days. For example, assume four client computers are selected to further evaluate the fitness of a gene stored in pool 14. Accordingly, the first selected client computer is instructed to evaluate the gene over a first time period; the second selected client computer is instructed to evaluate the gene over a second time period; the third selected client computer is instructed to evaluate the gene over a third time period; and the fourth selected client computer is instructed to evaluate the gene over a fourth time period. It is understood that the first, second, third and fourth time periods are different time periods that may or may not overlap with one another. Thereafter, the server receives the fitness

values from the selected client computers and combines these fitness results with the previous fitness value of the gene--as was maintained by the server prior to sending the gene back to the client--to arrive at an updated value for the gene's fitness value. Therefore, in accordance with the present invention, the speed at which the genes are aged is enhanced by distributing the evaluation task among a number of client computers operating in parallel. In one embodiment, the average of previous and new fitness values is used to compute a new fitness value for a gene that is sent to clients by the server for further evaluation. Since the genes in the server are sent to several clients for evaluation, only the results of partial evaluations of the genes are lost if one or more clients fail.

[0050] A backup/restore process for the server pool gene may be performed to ensure continuity in the event of the server failure. Moreover, because the clients are configured to have copies of the server genes they were instructed to evaluate and because the clients are self sufficient in their evolutionary process, the clients can continue evaluating their gens and advance the evolutionary process even when the server fails or is otherwise off line. When the server is back on-line, the server pool can even be recreated from genes stored in sent the clients. Therefore, a network computing system, in accordance with embodiments of the present invention does not lose the history of the prior processing of the genes.

[0051] Data feed server 50 provides historical financial data for a broad range of traded assets, such as stocks, bonds, commodities, currencies, and their derivatives such as options, futures etc. Data feed server 50 may be interfaced directly with server 20 or clients. Data feed servers may also provide access to a range of technical analysis tools, such as financial indicators MACD, Bollinger Bands, ADX, RSI, and the like.

[0052] The genes in the server pool may over time begin to behave similarly and use the same set of indicators in a correlated manner to arrive at their recommendations. This indicates that the search for fitter genes is converging on a set of identifiable conditions and indicators initially used to define the genes. As convergence occurs, the rate at which fitter genes in any give pool can be identified starts to diminish. In other words, as the genes age, their fitness increases, thereby leading to a higher convergence factor for the genes. Convergence may reach a point at which the surviving gene pool remains relatively unchanged despite further evaluations of the genes. Thus, convergence while required to identify fitter genes, can adversely affect the diversity of the gene pool. In the continuing

search for fitter genes, convergence may therefore represent a local optima and not an optimum point.

[0053] Figure 3 shows an exemplary convergence factor as a function of evaluation time for a given pool. It is seen that as the evaluation time increases, the rate of convergence asymptotically approaches a constant value of Q. A number of different techniques may be used to measure the degree of convergence and homogeneity of a gene pool. Expression (1) below provides one measure of the convergence factor of a gene pool:

$$\text{Convergence Factor} = \sqrt{\frac{\text{weighted sum of average fitness of the genes}}{\text{sum of average fitness of the genes}}}$$

where:

weighted sum of average fitness = weighted sum of average fitness + average fitness × age

[0054] In accordance with some embodiments of the present invention, the sever divides the gene pool into a number Z of different classes. Accordingly, the server divides the client computers into Z groups each being associated with a different gene class. For example, clients in group 1 are associated with class 1 genes; clients in group 2 are associated with class 2 genes; and clients in group Z are associated with class Z genes. Different groups may or may not have the same number of client. The server maintains a record of the class to which each gene and client belong. The genes in each class are characterized, in part, by a different subset of indicators that collectively and together with a multitude of conditions define the gene pool. Each class is thus enabled to contribute to a partial solution or make recommendations in a search space defined by the indicators associated with that class.

[0055] In some embodiments the server redistributes the genes for further evaluation only among clients that are members of the same class. Consequently, multiple classes of clients are run at the same time with a common server pool. For example, assume, that a sever has divided the client computers into three groups (classes) each being assigned to (i.e., being associated with or a member of) a different one of the three class of genes. The genes in the first class are enabled to use, for example, Tick-price, the trade volume and the volatility of the stocks as indicators. The genes in the second class are enabled to use, for example, Tick-price, MACD and ADX information as indicators. The genes in the third class are enabled to

use, for example, Tick-price, the trade volume, and the rate of change of the stock price as indicators. In accordance with such embodiments, each class operates independently and transfers its elitist pool to the server, as described above. To evaluate the genes further, the server sends the genes of each class only to client members of that class. For example, the elitist genes of class 1 identified by client members of class 1, can only be sent to the client members of class 1 for further evaluation by the server. Therefore, in such embodiments, genes belonging to different classes are not merged.

[0056] Figure 4 shows a network computer system 300 having a sever 310 and a multitude of clients 320_{ij} forming Z different classes, in accordance with one exemplary embodiment of the present inventions. Index i identifies the client number within each class and is shown to vary from 1 to N; index j identifies the class number and is shown to vary from 1 to Z. For example, client 320_{11} identifies the first client computer in gene class number 1; client 320_{21} identifies the second client computer in gene class number 1, and client 320_{1N} identifies the N^{th} client computer in gene class number 1. Similarly, client 320_{Z1} identifies the first client in gene class number Z, client 320_{Z2} identifies the second client computer in gene class number Z, and client 320_{ZN} identifies the N^{th} client in class number Z. It is understood that N and Z are integers, and that each class may have a different number N of clients.

[0057] In one embodiment, as described above, genes received by server 310 from any of class j clients, can only be sent back to class j clients for further evaluation. For example, genes received by the server from class 1 clients 320_{i1} (i.e., class 1 genes) can only be sent to class 1 client 320_{i1} for further evaluation. The classification of the genes, as described above, and their association with client computers may, as described above, result in gene homogeneity, faster convergence and absence of correlation between different gene classes.

[0058] In accordance with another embodiments of the present invention, the server merges the classes that have converged sufficiently to generate a new class of genes. To achieve this, the server is enabled to transfer the genes it receives from one member of a client class to member clients of a different class. Alternatively, the server may form a new class of genes by merging two or more existing classes. The new class is thus defined by combining the indicators used by the merged classes. For example, a new class generated by merging two different classes may be characterized by the union of the set of indicators used by the two merged classes.

[0059] Referring to Figure 4, assume that server 310 enables mixing of the genes. Accordingly, server 310 may send, for example, class 1 genes it receives from clients 320_{i1} only to other class 1 clients for further evaluation; or send, for example, class 4 genes it receives from clients 320_{i4} only to class 4 clients for further evaluation. Assume, for example, that the genes in the first class are enabled to use indicators Tick-price, trade volume and the volatility of the stocks, and the genes in the second class are enabled to use indicators Tick-price, MACD and ADX information. Assume further that the server merges the first and second class of genes to from a new class of genes. Accordingly, the indicators used by the newly generated class include Tick-price, trade volume, volatility, MACD, and ADX information. The merging of the genes helps to minimizes any convergence that may result from finding a local optima that is not the optimum point. The merged classes may or may not be discarded.

[0060] Figure 5A shows an exemplary flowchart 500 for evaluating performance characteristics of a number of genes by a multitude of client computers, in accordance with one embodiment of the present invention. During an initialization stage, the client computers contact the server to be assigned 502 by the server to one of N different classes each class being associated with and corresponding to one of N classes of genes. Accordingly, there are N classes of genes each associated with one of the N classes of client computers. Following the generation 504 of genes and receipt 506 of data associated with the genes, the genes are evaluated 508 using the received data to determine their performance characteristics or fitness. Following the evaluations 508, genes whose fitness are determined 510 as being less than a threshold value, are discarded 512. Genes whose fitness are determined 510 as being greater than or equal to the threshold value are stored and provided 514 for selection and acceptance by a server computer.

[0061] Figure 5B shows an exemplary flowchart 550 for evaluating performance characteristics of a number of genes by one or more server computers, in accordance with one embodiment of the present invention. In the example shown in Figure 5B, prior to accepting a new gene, the server computer determines 562 whether the new gene was previously accepted and stored by the server. If the server computer determines that the new gene was previously accepted and stored by the server computer, the server computer combines 564 the fitness value of the new gene with its old fitness value and accepts 554 the gene for storage. If the server computer determines that the new gene was not previously accepted and stored by the server computer, the server computer compares 552 the fitness value of each such gene

to the fitness values of the genes previously stored by the server computer. If this comparison 552 shows that the fitness value of a gene provided for acceptance is greater than or equal to the fitness values of the genes previously stored by the server computer, the server computer accepts 554 the gene. If this comparison 552 shows that the fitness value of the gene provided for acceptance is less than the fitness values of the genes previously stored by the server computer, the server computer may not accepts 560 the gene.

[0062] For every gene accepted by the server computer, the server computer determines whether the convergence factor of the gene class to which the accepted gene belongs meets 556 a specified threshold or condition. If the convergence factor of the gene class to which the accepted gene belongs meets 556 the specified threshold or condition, the gene class is merged 574 with another gene class that has been qualified for merging, thereby generating a new class. When the merger occurs, the server changes 568 the class of the genes that are merged to the new class. The class of the client computers of the merged classes is similarly changed. Once the class of the merged classes changes 568, a determination is made 558 about the time period used to evaluate the newly accepted gene. If it is determined 558 that a newly accepted gene meets the required duration condition, the server computer stores 570 the newly accepted gene together with its fitness value.. If it is determined 558 that a newly accepted gene does not meet the required duration condition, the server sends 566 the gene back to one or more client computers for further evaluation covering more time periods. In such embodiments, the gene is only sent back to clients that had previously evaluated the gene and provided the gene for acceptance to the server computer. In other words, server genes that were evaluated by class j computers are only sent 560 to class j computers for further evaluation, where j is a variable varying from 1 to N. If the convergence factor of the gene class to which the accepted gene belongs does not meet 556 the specified threshold or condition, the process moves to act 558 to determine whether the time period used to evaluate the newly accepted gene meets the required duration condition, as described above. It is understood that the order in which the various acts and decisions are shown in Figure 5B are only exemplary and may be varied without departing from the functions described above.

[0063] Figure 6 shows a number of components of the client and server computers of Figure 1. Each server or client device is shown as including at least one processor 402, which communicates with a number of peripheral devices via a bus subsystem 404. These peripheral devices may include a storage subsystem 406, including, in part, a memory subsystem 408 and a file storage subsystem 410, user interface input devices 412, user interface output

devices 414, and a network interface subsystem 416. The input and output devices allow user interaction with data processing system 402.

[0064] Network interface subsystem 416 provides an interface to other computer systems, networks, and storage resources 404. The networks may include the Internet, a local area network (LAN), a wide area network (WAN), a wireless network, an intranet, a private network, a public network, a switched network, or any other suitable communication network. Network interface subsystem 416 serves as an interface for receiving data from other sources and for transmitting data to other sources. Embodiments of network interface subsystem 416 include an Ethernet card, a modem (telephone, satellite, cable, ISDN, etc.), (asynchronous) digital subscriber line (DSL) units, and the like.

[0065] User interface input devices 412 may include a keyboard, pointing devices such as a mouse, trackball, touchpad, or graphics tablet, a scanner, a barcode scanner, a touchscreen incorporated into the display, audio input devices such as voice recognition systems, microphones, and other types of input devices. In general, use of the term input device is intended to include all possible types of devices and ways to input information to.

[0066] User interface output devices 414 may include a display subsystem, a printer, a fax machine, or non-visual displays such as audio output devices. The display subsystem may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), or a projection device. In general, use of the term output device is intended to include all possible types of devices and ways to output information.

[0067] Storage subsystem 406 may be configured to store the basic programming and data constructs that provide the functionality in accordance with embodiments of the present invention. For example, according to one embodiment of the present invention, software modules implementing the functionality of the present invention may be stored in storage subsystem 206. These software modules may be executed by processor(s) 402. Storage subsystem 406 may also provide a repository for storing data used in accordance with the present invention. Storage subsystem 406 may include, for example, memory subsystem 408 and file/disk storage subsystem 410.

[0068] Memory subsystem 408 may include a number of memories including a main random access memory (RAM) 418 for storage of instructions and data during program execution and a read only memory (ROM) 420 in which fixed instructions are stored. File storage subsystem 410 provides persistent (non-volatile) storage for program and data files,

and may include a hard disk drive, a floppy disk drive along with associated removable media, a Compact Disk Read Only Memory (CD-ROM) drive, an optical drive, removable media cartridges, and other like storage media.

[0069] Bus subsystem 404 provides a mechanism for enabling the various components and subsystems of the client/server to communicate with each other. Although bus subsystem 404 is shown schematically as a single bus, alternative embodiments of the bus subsystem may utilize multiple buses.

[0070] The client/server may be of varying types including a personal computer, a portable computer, a workstation, a network computer, a mainframe, a kiosk, or any other data processing system. It is understood that the description of the client/server depicted in Figure 3 is intended only as one example Many other configurations having more or fewer components than the system shown in Figure 3 are possible.

[0071] The above embodiments of the present invention are illustrative and not limiting. Various alternatives and equivalents are possible. Other additions, subtractions or modifications are obvious in view of the present disclosure and are intended to fall within the scope of the appended claims.

Claims:

1. A server computer system for use with a plurality of client computers each assigned to a respective class, comprising:

a memory accessible to the server computer system and storing a server candidate pool having a plurality of individuals, each individual identifying a plurality of indicators and at least one corresponding action in dependence upon the indicators, each of the classes being associated with a respective subset of indicators in a plurality of indicators, not all of the subsets being the same;

a communications port through which the server computer system receives individuals from the client computers, including a first subset of the individuals all received from a first one of the classes; and

a processor configured to:

determine whether the first class satisfies a predetermined convergence condition, and if so then to merge the first class with a second one of the classes to derive a merged class, at least some of the client computers that were assigned to the first class being re-assigned to the merged class, and

transmit to the client computers for further evaluation, each individual in the first subset of the received individuals;

wherein each individual in the first subset transmitted for further evaluation is transmitted for further evaluation to:

a client computer in the first class, if the first class did not satisfy the predetermined convergence condition, and

a client computer in the merged class, if the first class did satisfy the predetermined convergence condition.

2. The server computer system of claim 1, wherein the processor is further configured to:

determine whether each of the received individuals is already stored in the server candidate pool; and

discard all of the received individuals having an updated fitness value which is below a predetermined server minimum fitness, wherein the updated fitness value for each received individual is,

for individuals which are not already present in the server candidate pool, a fitness value received by the server computer system in conjunction with the received individual; and

for individuals which are already present in the server candidate pool, a combination of a fitness value received by the server computer system in conjunction with the received individual and a fitness value previously associated with the individual in the server candidate pool.

3. The server computer system of claim 2, wherein the predetermined server minimum fitness is dependent upon a fitness of the least fit individual in the server candidate pool.

4. The server computer system of claim 2, wherein the processor is further configured to store in the server candidate pool at least some of the accepted individuals.

5. The server computer system of claim 1, wherein the individuals received by the server computer system through the communications port further include a second subset of individuals all received from the second class,

wherein the second class is also determined by the server computer system to satisfy a predetermined convergence condition,

wherein the merging of the first class with the second class further includes reassigning to the merged class at least some of the client computers that were assigned to the second class,

wherein the processor is further configured to transmit to the merged class of client computers for further evaluation, each individual in the second subset of the received individuals.

6. The server computer of claim 1, wherein the merged class is associated with a union of the subset of indicators previously associated with the first class and the subset of indicators previously associated with the second class.

7. The server computer system of claim 1, wherein one of the individuals transmitted for further evaluation is transmitted for further evaluation also to:

an additional client computer in the first class, if the first class did not satisfy the predetermined convergence condition, and

an additional client computer in the merged class, if the first class did satisfy the predetermined convergence condition.