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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 May 2003 (08.05.2003)

PCT

(10) International Publication Number
WO 03/038749 A1

(51) International Patent Classification[7]: G06N 5/02

(21) International Application Number: PCT/US02/34571

(22) International Filing Date: 28 October 2002 (28.10.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/335,654 31 October 2001 (31.10.2001) US

(71) Applicant *(for all designated States except US)*: **ICOSYSTEM CORPORATION** [US/US]; 545 Concord Avenue, Cambridge MA 02138 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BRANKE, Juergen** [DE/DE]; Ernststrasse 17, Karlsruhe 76131 (DE). **CAMPOS, Michael** [US/US]; 3007 Garcia Walk, Los Angeles, CA 90026 (US).

(74) Agents: **MORRIS, Francis, E.** et al.; Pennie & Edmonds LLP, 1155 Avenue of the Americas, New York, NY 10036 (US).

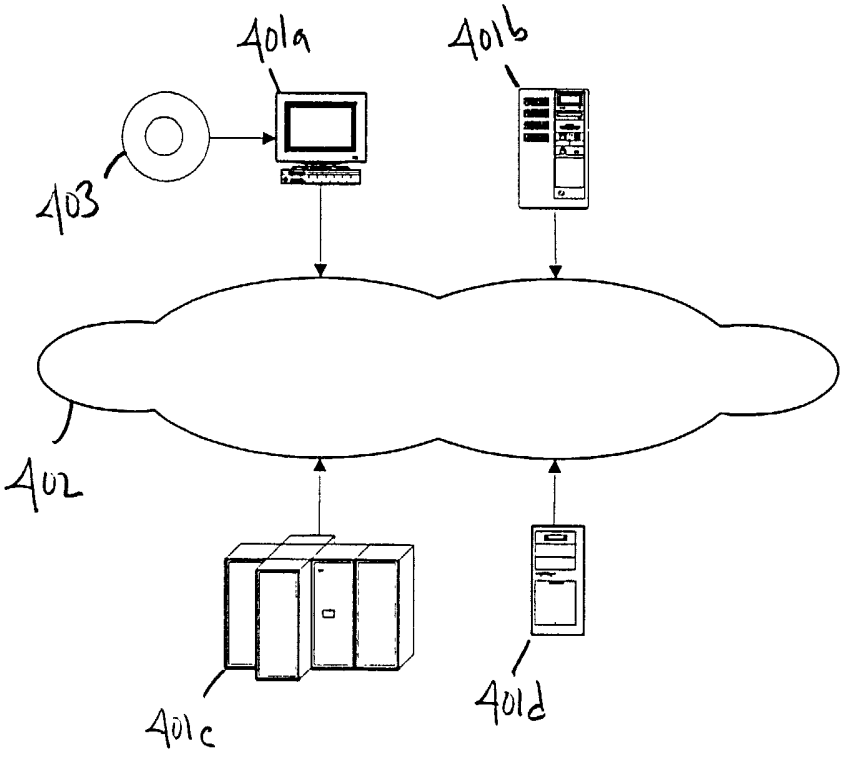
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR IMPLEMENTING EVOLUTIONARY ALGORITHMS



(57) Abstract: A method that efficiently implements evolutionary algorithms on heterogeneous computer clusters (Fig. 4); in which method a central process delegates subpopulations of individuals of similar fitness (211) to separate computers where they evolve for a certain number of generations (212) after which they return to the central pool before the delegation is repeated.

WO 03/038749 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND SYSTEM FOR IMPLEMENTING EVOLUTIONARY ALGORITHMS

FIELD OF THE INVENTION

The present invention relates to methods and systems for implementing evolutionary algorithms in a parallel computing environment. More particularly, the invention efficiently performs evolutionary algorithms on computers interconnected via an intranet or the internet.

BACKGROUND OF THE INVENTION

Evolutionary algorithms (EA) have come to represent an important paradigm for solving or approximating the solutions to combinatorial optimization problems, especially hard optimization problems, e.g., the traveling salesman's problem. See, e.g., Zbigniew Michalewicz, Genetic Algorithms & Data Structures = Evolution Programs (Springer 1999).

To use the EA paradigm for a particular problem requires that possible solutions to the problem be parameterized by a data structure capable of the "genetic" operations to be described, and that the quality of the solution in the problem be represented by an objective (or "fitness") function applied to the possible solution. Genetic algorithms (GA) are special cases of EAs where the data structure parameter is a fixed-length list of values. In classical GAs, the values are single bits. The data structure parameter defining a possible solution is known as a "chromosome," and the individual elements of the data structure are known as "genes."

Generally, EAs search for increasingly good (or "fit") solutions, or even an optimal solution, by performing a number of repeated transformations or iterations on a collection of possible solutions. Each possible solution is known as an "individual"; the collection of possible solutions is known as a "population"; and each iteration is known as a "generation." Generally, the number of individuals in a population is constant from generation and is an important EA parameter.

In more detail, each generation includes four basic steps. In a first step, the individuals are selected from the general population for the next step. Selection of individuals may require global information to determine which individuals are chosen from the population. In the classic selection method, individuals are probabilistically selected

based on their relative fitness with respect to the total fitness of the population. Thus, the higher the fitness, the more likely the individual would be selected for the next step. In another selection method, individuals are probabilistically selected based on fitness rank within the entire population (rank-based selection), or on fitness rank within a random sample of the population (tournament selection). In a further alternative, the more fit individuals in the population are selected (elitist selection).

Next, in a second step, the selected individuals are altered to form new individuals by performing genetic operations. In most cases, the genetic operators include "mutation," in which part of the data structure parameterizing an individual is randomly changed, and "crossover," in which portions of the parameterizing data structures of two individuals are exchanged. The frequencies of mutation and crossover (and how individuals are chosen for crossover) are further EA parameters. For example, in classical GAs where the data structure parameter is a bit string, mutation may flip one bit, while crossover may exchange portions of two strings.

In the third step, the fitness of all the new individuals in the population at the current generation is determined by applying the fitness (or objective) function.

In the last step, the individuals that will comprise the succeeding population are determined. The succeeding population is ordinarily formed in one of two ways. In generational reproduction, each new individual created by the second step replaces one of the parents, sometimes keeping the best parent (elitism). Alternatively, the generational reproduction method requires that the new individual must be better than both parents in order to remain, otherwise both parents remain. The other method is known as steady-state reproduction. In steady-state reproduction, both parents are retained, and the fitness evaluation performed in the third step determines which individuals in the whole population are the least fit. These least fit individuals are removed from the population, thus reducing the population to its original size.

An initial population of individuals may be created by simply randomly assigning content to the parameterizing data structure (the chromosomes). For classic GAs, bits in the bit string may be randomly set to zero or one.

Because EAs operate on populations of semi-independent individuals, they offer many opportunities for parallelization. In a simplest parallelization technique, computation of the fitness function for each individual, which is often the most computationally intensive part of an EA, may be done in parallel. Additional parallelization techniques are known in the art. See, e.g., Erick Cantu-Paz, Efficient and accurate parallel genetic algorithms (Kluwer Academic 2000), and Schmeck et al., Parallel implementations of evolutionary algorithms, in Solutions to Parallel and Distributed Computing Problems 47-

68 (Zomaya et al. eds., Wiley 2001). In particular, the following three parallelization techniques are known.

- **The farming model:** Here, the population selection step is performed on only one processor, while pairs of individuals are distributed to other processors for genetic alteration and the usually time-consuming fitness evaluation. This model is suited for systems with a small number of processors.
- **The island model:** In this model, the population is divided into a number of "subpopulations." Subpopulations are assigned to separate processors where they evolve independently, except for occasional exchanges of individuals between neighboring subpopulations. Subpopulations are neighboring if they are adjacent according to a chosen communication topology, for example a ring interconnection of the subpopulations. This model is suited to clusters of workstations, each powerful enough to process a whole subpopulation, and with relatively high bandwidth communication connections.
- **The diffusion model:** Here, a spatial distribution is defined on all the individuals in the population, and selection is restricted to only those individuals in a local neighborhood. This model is suited to a highly parallel single instruction stream, multiple data stream computer, with each individual being assigned its own processor. Communication in this model is purely local, and the computational requirements on each node are limited.

The latter two techniques are not formally exactly equivalent to the standard EA paradigm. Instead, in these latter models, the selection process has been distributed and localized (instead of operating on the population as a whole) in order to reduce inter-processor communication.

However, the known parallelization techniques for EAs have been developed with particular parallel computing platforms in mind. In particular, these techniques expect that both processor and also inter-processor communication capabilities are preferably homogenous, or at least known in advance. Such expectations are met by, for example, dedicated massively parallel systems, or by dedicated local clusters of workstations on a local high-speed interconnection.

But some of the most ubiquitous parallel systems available currently, intranets or the Internet, meet none of these expectations. In particular, they have heterogeneous capabilities, have unpredictable processor availability, and have unpredictable and generally low communications bandwidth.

- **Heterogeneous capabilities:** Usually, the computers connected to these networks are of widely differing capabilities, ranging from simple PCs to powerful workstations or mainframes.
- **Unpredictable availability:** The connected computers may be otherwise used, and spare computing power available for other tasks, such as EA processing, is therefore unpredictable.
- **Low and unpredictable bandwidth:** Different computers have vastly different network connections, ranging from direct connection to backbone to a slow analog modem. Overall, the connections are slow compared to connections within local clusters of computers.

None of the known parallelization techniques are suited to these most common examples of parallel systems.

For example, low and unpredictable communication bandwidth severely limits the use of both the farming model and the diffusion model, since these models have communication demands that require high and predictable bandwidth. Further, subpopulations that are local according to the selected topology may reside on real processors that are a considerable distance apart, further burdening the communications connections.

On the other hand, use of the island model is also severely limited by heterogeneous and unpredictable processor capabilities. Variable processor capabilities lead to highly uneven evolution of the separate subpopulations. Subpopulations on slower processors will evolve less rapid than subpopulations on a faster processor. Then, when individuals migrate from slower to faster processors, the immigrants will be generally less fit than the natives, and will be rapidly eliminated by selection. Thus, computing resources of the slower processors is effectively wasted. See, e.g., Branke, et al., A distributed genetic algorithm improving the generalization behavior of neural networks, in Machine Learning: ECML-95, 912 LNCS 107-21 (N. Lavrac and S. Wrobel, eds., Springer 1995).

What is needed, and what is lacking in the prior art, is a technique that exploits the natural parallelism in EAs in networks of heterogenous computers with heterogeneous and generally slow communications bandwidth.

SUMMARY OF THE INVENTION

Objects of the present invention include overcoming these problems in the prior art. Accordingly, this invention provides an efficient and effective implementation of general evolutionary algorithms (EA) as a number of communicating and cooperating processes,

which is particularly suited for heterogeneous computer networks such as intranets, the Internet, or the like. Especially, this invention enables exploitation of otherwise unused computing power of networked computers for application of EAs to large, real-world optimization problems.

These objects are accomplished by setting aside one, distinguished processes (designated herein the "central" process) to maintain a pool ("central pool") of "individuals" for distribution to, and "evolution," by the other processes (designated herein the "peripheral" processes). The designations "central" and "peripheral" are to be understood functional labels used without any other limitation intended. In particular these terms are to be understood as carrying no implications of spatial arrangement, geographical allocation, or so forth. For example, the central and peripheral processes may be physically located on a single processor machine, a machine with multiple processors, or they may be distributed among several machines.

When a peripheral process is available for work, it notifies the central process, which then provides it with an allocation of individuals for evolution according to the EA being implemented. The peripheral process then evolves these individuals through a number of "generations," then returns the evolved population to the central process, and then again requests additional work.

When the central process receives a request for work from a peripheral process, it migrates a collection (also called a "subpopulation") of individuals to the peripheral process. The individuals picked most preferably have fitness values forming a contiguous range (or, preferably, relatively similar fitness values) with respect to the fitness values of all the individuals in the central pool. When the central process receives an evolved population from a peripheral process, the evolved individuals are merged into the pool. For the migration and merging, it is advantageous for the central pool to be maintained in an order sorted according to fitness. A sorted central pool is called herein a "central buffer."

Advantageously, this invention limits competition between individuals of widely different fitness by distributing subpopulations selected to have relatively similar fitness to the distributed processes. Individual selection and subpopulation evolution occur only in the peripheral processes. Communication between the processes is reduced because subpopulations evolve for several generations in the peripheral processes without any intervention by the central process.

The invention consists generally of a method of performing an evolutionary algorithm on a network of heterogeneous computer processors, comprising the steps of creating a set of subpopulations of individuals, processing each subpopulation with the evolutionary algorithm until a termination condition is reached, migrating individuals so

that each subpopulation contains individuals of relatively similar fitness, and, repeating the processing and migrating steps until an ending condition is reached.

One aspect of the invention refines the migrating step by merging the incoming subpopulation into a central pool of individuals ordered by fitness and transferring an outgoing subpopulation of relative similar fitness from the central pool. Other features include keeping the number of individuals in each subpopulation equal, or alternatively sizing the number of individuals in each subpopulation relative to the processing capability of the processor. More features include performing all steps on a single processor, or alternatively using one or more peripheral processors to perform the processing step, and locating each subpopulation on a peripheral processor.

Another aspect of the invention refines the migrating step by merging the incoming subpopulation into an intermediate pool of individuals ordered by fitness and transfers an outgoing subpopulation of relative similar fitness from the intermediate pool, and includes an additional step of exchanging individuals between intermediate pools with a central pool after a certain number of subpopulation migrations.

The invention also includes a computer program storage media, containing encoded software instructions including an evolutionary algorithm, wherein the encoded instructions cause one or more computer processors to create a set of subpopulations of individuals, process each subpopulation with the evolutionary algorithm until a termination condition is reached, migrate individuals so that each subpopulation contains individuals of relatively similar fitness, and, repeat said process and migrate steps until an ending condition is reached.

Citation or identification of any reference in this Section or any section of this application shall not be construed that such reference is available as prior art to the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention may be understood more fully by reference to the following detailed description of the preferred embodiment of the present invention, illustrative examples of specific embodiments of the invention and the appended figures in which:

Figure 1 depicts an embodiment of the functional structure of the present invention;
Figures 2A and 2B depict flow charts of the methods of this invention;
Figure 3 depicts a preferred embodiment of selecting and merging individuals;
Figure 4 depicts an exemplary computer system suited to the present invention;

Figure 5 depicts a comparison of the invention with an island model; and,
Figure 6 depicts a comparison of the invention with an optimized island model.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

As a preliminary matter, although the invention is described herein as, inter alia, a method for implementing EAs, it is equally applicable to implementing other similar algorithms, such as genetic algorithms. One of skill in the art will recognize from the following description that the invention may be used to implement algorithms that maintain a population of intermittently interacting, independent individuals that have associated fitness values or ranks.

Further, although the invention is also described as including communicating processes which carry out work requests, these processes are functional descriptions without any limitation intended concerning implementation in any particular operating system (OS). For example, in one implementation, the functional processes described are implemented as actual OS processes, while in another implementation no such OS processes may be discernible. In the latter case, the process functions may be performed by native OS components and may be triggered by message exchanges. Such variations as may be necessitated due to detail of a particular OS and particular hardware are therefore within the scope of the present invention.

Also, for concreteness of description only, and without any limitation, the preferred embodiments are described herein using the notation introduced above along with the further notation. The EA being implemented is considered to have M individuals (designated $d_1, \ldots, d_M$); the method is considered to have N communicating functionally-defined processes (designated $p_1, \ldots, p_N$).

Both M and N may be variable during the method, as will become clear from the following description. Anticipating the following description, the number of processes, N, may vary, because, since the central process passively responds to peripheral processes, upon creation a new peripheral process will simply request additional work like the already created peripheral processes. The number of individuals may easily be varied, because new individuals may be simply merged into the central pool (subject resource constraints), while individuals may be destroyed by simply deleting them from the central pool.

Now, Fig. 1 illustrates a preferred embodiment of the functional structure of the present invention, which includes communicating processes 101a-101d that cooperate to implement a wide range of preselected evolutionary algorithms (EA). The implemented EA may be selected from a virtually unlimited variety of evolutionary algorithms (or

structurally similar algorithms) virtually any nature or complexity as known in the art and directed to the solution of a combinatorial optimization problem (or structurally a similar problem) of virtually any type. See, e.g., Solutions to Parallel and Distributed Computing Problems (Zomaya at al. (eds.)) (describing the use of EAs for a wide range of problems).

Preferably, but not necessarily, one of the processes, for example process 101a, is designated as a central process, while the remaining processes, processes 101b-101d, are designated as peripheral processes. All processes communicate through interconnection 102, which may be any type of a local or remote interprocess communication mechanism. The functional topology of interprocess communication is a "hub-and-spoke" with the central process being the hub and the peripheral processes being the spokes. The actual physical topology within interconnection 102 is not otherwise limited.

Figs. 2A and 2B generally illustrate preferred embodiments of the functional processing of a central process and of a peripheral process, respectively, which cooperate to implement a preselected EA. With reference first to Fig. 2A, the central process starts and initializes itself at 201, and then proceeds to test 202 how many individuals are currently in its central pool of individuals. If the current number of members is adequate, then the central process proceeds to wait 204 for a work request from a peripheral process. If the pool size is inadequate, additional individuals are initialized 203. Individual initialization typically may require finding the fitness of the newly generated individuals. Alternatively, the new individuals may simply be assigned an average (or a minimum) fitness.

Initially, the pool will be empty and an initial number of individuals may be initialized, where the initial number may be, for example, an input parameter of the method. Later during processing, it is advantageous to dynamically maintain the pool size so that it is unlikely that a work request from a peripheral process must be delayed because there are insufficient, or no, individuals in the pool. If a delay occurs, then in one embodiment the number of individuals (and perhaps the maximum possible pool size) may be increased. This could be achieved by generating newly initialized individuals, e.g. either randomly (but if random, preferably during the preliminary generations of the execution of the algorithm) or by duplicating some individuals. If the pool becomes too full, then resources may be allocated to increase the maximum pool size. Alternatively, individuals could be removed from the pool to clear space, e.g. by removing the least fit individuals.

For a concrete example, let the average number of individuals processed by each peripheral process at a certain time be P, so that M (the total number of individuals) = N (the total number of processes at a certain time) * P. Then it is preferred that the maximum capacity of the pool be approximately (within 10%) M individuals. And at any time during

processing, it is preferred that the pool be from 40-60% full, or less preferably from 30-70% full, or have at least P individuals.

Individuals are initialized 203 preferably by random generation or by taking into account some prior heuristic knowledge. In random generation, elements of the chromosomal data structure parameterizing an individual are randomly selected according to a probability distribution. Possible probability distributions are the uniform and the Gaussian distribution. For example, for a chromosome which is a bit string, as for classical GAs, the bit may be randomly set as zero or one with equal probability. Although Fig. 2A illustrates that individuals are initialized by the central process, in an alternative implementation, the central process may instruct a peripheral process to initialize individuals for its immediate use (thereby avoiding some communication between the processes).

If the current pool size is adequate, then the central process waits 204 for the next work request from a peripheral process. When a work request arrives, the evolved subpopulation just processed by the peripheral process is merged 205 into the central pool. In case the peripheral process has just started, it will have no evolved subpopulation to return, and the central process may bypass merging 205. Peripheral processes will startup during initialization, and may occasionally startup and join the ongoing processing even after initialization. The merging and migrating operations are described subsequently.

Next, the current status of the processing is evaluated to determine if one or more stop criteria have been met 206. If so, processing is stopped 208. Stopping may involve, for example, stopping immediately upon meeting the stop criteria, or waiting for all peripheral processes to finish performing the EA and to return their evolved subpopulations, or waiting until most (75%) do so, or waiting a fixed time for most to do so, or so forth.

However stopping is performed, the contents of the central pool after stopping is completed defines the results of the EA implemented according to the present invention.

If processing is to continue, the central process migrates 207 a new subpopulation to the peripheral process so that it may be evolved according to the EA being implemented. Finally, the central process again tests the current size of the central pool and waits for the next peripheral-process work request.

Next, Fig. 2B generally illustrates the processing of any of the peripheral processes. A peripheral process starts and initializes itself at 210, and then proceeds to request 211 a new subpopulation of individuals to evolve. Preferably, the central process will select and send 207 a new subpopulation from the central pool with a delay determined principally by the inter-process communication overhead.

Next, the peripheral process processes 212 the subpopulation according to the preselected evolutionary algorithm (EA). The general steps performed are fitness determination, genetic alteration, and selection as previously described. Preferably, the peripheral process includes a local subpopulation pool that holds the individuals of the subpopulation, and the EA evolves the individuals in this local pool. Since the processing of subpopulations can occur independently of other subpopulations, such processing may be performed concurrently, on two or more processors either along with or separate from the central process.

Processing 212 continues until a termination condition is reached. In a preferred embodiment, this termination condition may be defined in terms of the EA being implemented. For example, the processing may be terminated after a predetermined number of generations have occurred, the predetermined number optionally depending on the relative number of individuals in the subpopulation. Alternatively, the processing may be terminated after the fitness level in the subpopulation has reached a level where it is not significantly improving, i.e., relatively unchanging (for example a relative improvement of less than 0.1% - or 0.5%, or 1%, or so forth - per generation).

After the EA processing step reaches the termination condition, the subpopulation is returned 213 to the central process for merging 205 into the central pool. Because a previous migration step transferred the same number of individuals in the outgoing subpopulation from the central buffer, the size of the central buffer remains fixed.

Finally, the peripheral process determines whether or not to stop 214. Peripheral-processes may be notified to stop by the central process, or may stop themselves after using a predetermined amount of processor resources, or may be stopped by a user of the processor on which the process executes, or so forth.

Next, the central pool of individuals and the operations of migrating subpopulations from and merging subpopulations into the central pool are described. In a preferred embodiment, illustrated in Fig. 3, the individuals in the central pool are maintained in an order sorted (or ranked) according to fitness. Accordingly, the individuals, for example, individual 304, present in central pool 302 at a typical moment during the method of the present invention are arranged with the most fit at the top of the illustrated pool, and the progressively less fit arranged progressively further down in the illustrated pool. (Available free space in the pool is not illustrated.) As described above, the central pool is preferably maintained with (0.4-0.6)*(M) individuals, where M is the total number of individuals being processed. Local pools 301 and 303 of peripheral processes (perhaps the same peripheral process) are illustrated with a smaller number (P) of individuals.

The operation of migration 306, first, selects a subpopulation of individuals from the central pool chosen on the basis of their fitness relative to all individuals in the central pool when chosen, and, second, sends the selected subpopulation (via the inter-process communication facilities) to a local pool 303 of a peripheral process, where it is evolved according to the preselected EA. This operation is performed upon request from a peripheral process after the peripheral process completed evolving its current subpopulation and is thus free for more work. If necessary, the peripheral process may queue at the central processor until a new subpopulation is available.

Preferably, the subpopulation has similar relative fitness. In one alternative, a subpopulation may be selected according to a template or randomly from a more numerous collection of individuals forming a contiguous range of fitness in the central pool. For example, a subpopulation of P individuals may be selected as every other individual from 2.0*P individuals with contiguous fitness, or randomly from 1.5*P individuals with contiguous fitness, or so forth. In another alternative, groups of individuals with superior fitness may be dispatched from the pool more frequently than those with poorer fitness. In yet another alternative, the processing speed of the hardware running the peripheral process will determine the average fitness of the group selected. Less preferably, a subpopulation is selected randomly from the whole central pool, or is selected by any of the means described for selecting individuals for alteration in the EA (classic, elitist, or tournament, etc.).

More preferably, the subpopulation has a contiguous range of relative fitness with respect to all the individuals in the central pool. Thus, Fig. 3 illustrates subpopulation 307 having a contiguous range of relative fitness in central pool 304 being selected and sent to local pool 303 in a peripheral process. The contiguous fitness range selected in the preferred embodiment may itself be chosen in several manners. In one manner, it is selected in a round-robin order by fitness rank so that all individuals receive a uniform chance for further evolution. In another manner, it may be selected in a priority order by fitness so that the most fit individuals received the greatest chance for further evolution. It may also be selected randomly with respect to fitness, or by a combination of several methods.

The operation of merging 305 receives a subpopulation of individuals that have finished evolution in the local pool 301 of a peripheral process and inserts them back into central pool 304. If the central pool is maintained in fitness order, a subpopulation is merged with individuals in the central buffer, thus retaining the fitness ordering. If the central buffer is empty, the subpopulation is merely inserted into the central buffer in fitness order. Preferably to improve merging efficiency, individuals in the subpopulation are ordered by their relative fitness in the local pool by the peripheral process before being sent

to the central process. Here, Fig. 3 illustrates a subpopulation in priority order in a local pool, being merged 305 in fitness order into central pool 302. The separate arrows 305 illustrate where the separate individuals from the local pool fit into the central pool in fitness order.

It is also advantageous to set aside a copy of the most fit individual discovered so far in a location separate from the central pool. Alternatively, copies of the most fit few (for example, five to ten) individuals may be stored

Although, the central pool has been described as maintained in the preferred fitness order, other data structures are possible, preferably data structures that make selection based on priority and merging efficient operations. For example, the central buffer may be maintained as a "fitness-ordered" heap. Less preferably, the central pool may be stored in any fashion, for example as an unordered set.

Finally, it will be apparent that this method is not critically affected by failure of a peripheral process. Failure of a peripheral process merely results in loss of the individuals it is currently evolving and the accumulated results of their prior optimization. The central process may compensate for failure by simply initializing new individuals to make up for those lost, and sending them for evolution when requested. Also, a new peripheral process may easily be started to replace a failed process (or simply to increase the throughput). The new peripheral process will request a subpopulation from the central pool and then begin evolving the requested subpopulation according to the EA being implemented.

It is also to be understood that such variations in the methods and data structures described herein as will be apparent to one of skill in the art upon reading the present specification are included within this invention and within the scope of the appended claims.

ALTERNATIVE EMBODIMENTS

The present invention has alternative embodiments, several of which are described herein. In a first alternate embodiment, the size of the subpopulation selected and sent to a peripheral process may be varied in dependence on the processing capability of the processor hosting the peripheral process. Such variation takes into account any heterogeneity of the processors executing the methods of the present invention, and permits relatively equal number of generations to occur on each distributed processor and for each subpopulation. This requires knowledge of the relative processing ability of the distributed processor hosting a peripheral process before selecting a subpopulation size. Also, the central pool will preferably have resources that are capable of holding all the

subpopulations at once, and sufficient individuals should be initialized to insure all distributed processors may be simultaneously active.

This invention may use any of the wide range of stopping criteria known in the EA arts appropriate for the EA being implemented. In a preferred embodiment, the method stops when the total number of evolution cycles reaches a predetermined limit. In an alternate preferred embodiment, the method stops when measures of the fitness of the current population cease to improve significantly (for example, by less than 0.05% - or 0.1%, or 0.5%, or 1% - for each new generation of the population). Fitness measures that may be used include the maximum fitness discovered to the current time, the maximum fitness at the current time, the average fitness, the minimum fitness, or so forth.

In a further alternate embodiment that reduces communications costs at the central process, the processes of the invention may be structured into a preferably two-level tree (more levels are possible) instead of a hub-and-spokes arrangement (or a one-level tree). The central process is at the root of the tree; intermediate processes are at the intermediate nodes of the tree; and peripheral processes are at the leaves of the tree. Each intermediate process has an intermediate pool of individuals, and acts like the "central process" described above with respect to its dependent peripheral processes But when individuals in an intermediate pool have performed a certain number of evolutions (or have processed for a certain time, or so forth), the population contained in the intermediate pool may be returned to the central process and merged into the central pool and a subpopulation, preferably of similar relative fitness, would be selected and sent to the intermediate process.

In yet another alternative embodiment, the central pool maintains only the minimum data concerning the individuals with detailed data and code being retained at the peripheral processes. For example, the central pool may maintain only the identity and fitness of each individual. In this embodiment (preferred aspect), the central processor maintains a list of the identity of individuals in the entire population ordered by fitness, and periodically signals the distributed process to stop and return the identity and fitness of individuals in the evolved subpopulation to the central pool. Optionally, the peripheral processes may notify the central process of their progress in order to account for processor heterogeneity. Upon selecting a new subpopulation, the central process sends only the identities and fitness to a requesting peripheral processor. The peripheral processor obtains the remaining data and code defining the individuals by directly requesting such from the peripheral processors where each individual currently resides (but are not being evolved).

Conceptually, this requires functional communication structured as a complete graph among all the processes, the central process and the peripheral processes. Since the communications between distributed processors would be intense during the exchange

migration period, in an intranet environment a device like a switch that provides separate communications paths between nodes would be preferable to a hub which provides only one path between nodes at a given time.

This latter alternative is exemplary of further embodiments where the functions of the central process are performed in a distributed manner by the peripheral processes. Methods of function distribution and parallelization can be used that are known in the art. These alternatives are typically not subject to failure of the central process, which critically affects distributed implementations.

SYSTEMS AND PROGRAM PRODUCTS OF THE INVENTION

The cooperating functional processes of this invention may advantageously be executed on a parallel computer system. Fig. 4 depicts an exemplary system for executing the processes of the present invention. Here, computers 401a-401d (of standard types and constructed as known in the art) are connected for intercommunication by network 402. The illustrated computers are of heterogeneous types, such as PC computer 401a, workstation computer 401d, mainframe computer 401c, and minicomputer 401b. Other types of computers, such as massively parallel computers may also be used. Network 402 may be of a wide variety of types, including a machine bus, an intranet, the Internet, or an analog or a packet switched telephone network.

The processes, as illustrated in Fig. 1, may be conveniently assigned to these computers in any way, but will preferably be assigned to balance computer loads and to maximize the throughput. For example, several processes may be assigned to powerful computers, such as mainframe 401c, a few processes to computers of intermediate capability, and only one or two to less powerful personal or workstation type computers, 401a and 401d.

The central process may be assigned to any computer, but is preferably assigned to that computer used by the user for controlling the method of this invention. Thus, it may be assigned to PC computer 401a.

In a preferred embodiment, processes of this system will use background resources of the various computers that are not otherwise used. For example, arrangements are known that permit any Internet-attached computer to perform background work on an as available basis for a distributed processing task. This invention may be used with arrangements. See, e.g., Parabon Computation, Inc. (Fairfax, VA; www.parabon.com).

Additionally, a further embodiment of this invention is computer software stored upon media that will enable a computer such as PC computer 401a, workstation computer 401d, mainframe computer 401c, and minicomputer 401b that reads storage media 403 and

loads the software to perform the method of the invention as previously described. This software can be stored on any computer readable media such as, but not limited to, magnetic disk or tape, or optical media such as a CD ROM.

AN EXAMPLE OF THE INVENTION

The following example demonstrates the effectiveness of the present invention and its superiority to prior art parallelization techniques, e.g., the island model.

To execute the methods of the present invention and the island model, a distributed computing environment of ten independent and heterogeneous processors was simulated on a single processor computer. Time was measured by the number of generations, so that the overheads of selection, reproduction or mutation, and communications were neglected. Processing speed was simulated in terms of the number of "global cycles" a processor needs to process one generation. The simulated processors ranged from one to ten global cycles, or a 10-fold speed range.

The EA selected was directed to optimizing f(y), which was defined by the two following equations.

$$x_i = \begin{cases} y_i\cos(\pi/6) - y_{i+1}\sin(\pi/6) & : \ i=0,2,4,6,8 \\ y_i\cos(\pi/6) + y_{i-1}\sin(\pi/6) & : \ i=1,3,5,7,9 \end{cases}$$

$$f(y) = \sin(x_i) * \sin(\frac{i+1}{\pi} x_i)$$

(This problem was suggested by Michalewicz and used in a contest at the Evolutionary Computing Conference in 1999). Each individual was defined by a gene that included a list of the 10 real values: y_i, i = 0, ..., 9. The selected EA was further defined in a standard manner using linear rank-based selection, steady-state reproduction, mutation probability of 0.1 for each gene, and crossover probability of 1.0. The subpopulation size was 50, and there were 10 subpopulations for a total of 500 individuals. These settings were used both for the methods of this invention and for the island model, so that processing of the invention and of the island model was performed under the same conditions.

Further settings used for the first experiment are described next. For the invention, the central pool size was set to 500, and a subpopulation was allowed to evolve for 20 generations in each peripheral process until it was returned the results to the central pools. For the island model, a ring topology was used, with a migration interval of also 20 generations. Each experiment was run 50 times using different seeds.

In Figs. 4 and 5, illustrating experimental results, solution quality (fitness) increases along the vertical axis, and the number of generations increases along the horizontal axis.

Fig. 5 illustrates the results of the first experiment on the heterogenous system simulated, where curve 501 are the results of the methods of this invention, and curve 502 is the result of the island model. The present invention clearly reaches a final solution that is considerably over the solution of the island model, although this invention does converge somewhat slower.

In a next experiment, the EA parameter settings were tuned to optimize performance of the island model regardless of any deleterious effect on the methods of this invention. Optimized results for the island model were obtained with a subpopulation size of 400 (not 50) and a migration interval of 40 (not 20) generations. The same parameters were used for the methods of this invention.

Fig. 6 illustrates the results of this second experiment, where curve 601 are the results of the methods of this invention, and curve 602 is the result of the island model. Even compared against an optimized island model, the present invention converged to a better final solution than the island model.

This example clearly demonstrated the usefulness and effectiveness of the present invention and its superiority to the island model.

The invention described and claimed herein is not to be limited in scope by the preferred embodiments herein disclosed, since these embodiments are intended as illustrations of several aspects of the invention. Any equivalent embodiments are intended to be within the scope of this invention. Indeed, various modifications of the invention in addition to those shown and described herein will become apparent to those skilled in the art from the foregoing description. Such modifications are also intended to fall within the scope of the appended claims.

A number of references are cited herein, the entire disclosures of which are incorporated herein, in their entirety, by reference for all purposes. Further, none of these references, regardless of how characterized above, is admitted as prior to the invention of the subject matter claimed herein.

CLAIMS

What is claimed is:

1. A method for performing an evolutionary algorithm comprising a fitness function and an ending condition, the method comprising:

selecting one or more subpopulations of candidate solutions of relatively similar fitness values from a pool of available candidate solutions,

processing the subpopulations according to the evolutionary algorithm until a termination condition is reached,

merging subpopulations whose processing has terminated back into the pool, and,

repeating said selecting, processing, and merging until the ending condition of the evolutionary algorithm is reached.

2. The method of claim 1, wherein available candidate solutions selected from the pool have relatively similar fitness values if their fitness values form a contiguous range among the fitness values of all available candidate solutions in the pool.

3. The method of claim 1, wherein each subpopulation has the same number of candidate solutions.

4. The method of claim 1, wherein each subpopulation is processed on a processor, and wherein subpopulation size is selected in dependence on the capability of the processor.

5. The method of claim 4 wherein the subpopulations are processed on two or more processors of heterogenous capabilities.

6. The method of claim 1 wherein processing a subpopulation comprises one or more generations, each generation comprising:

selecting candidate solutions from the subpopulations,

creating new candidate solutions by applying genetic operators to the selected solutions, and

evaluating the fitness of created solutions.

7. The method of claim 6 wherein the termination condition comprises processing for a fixed number of generations.

8. The method of claim 6 wherein the termination condition comprises processing until fitness of new candidate solutions ceases to increase.

9. The method of claim 1, wherein at least part of the processing of at least two subpopulations is concurrent in time.

10. The method of claim 1, wherein all steps are performed on a single processor.

11. The method of claim 1, wherein a central processor maintains the pool.

12. The method of claim 11, wherein one or more peripheral processors of heterogeneous capabilities and communicatively linked to the central processor process the subpopulations.

13. The method of claim 1, wherein membership in the pool is determined by communications between two or more processors.

14. The method of claim 1, wherein merging merges the incoming subpopulation into an intermediate pool of candidate solutions ordered by fitness and transfers an outgoing subpopulation of relative similar fitness from the intermediate pool, and includes an additional step of exchanging candidate solutions between intermediate pools with the pool after a certain number of subpopulation merges.

15. A computer program storage media comprising encoded software instructions for causing one or more computers to perform the steps of:

selecting one or more subpopulations of candidate solutions of relatively similar fitness values from a pool of available candidate solutions,

processing the subpopulations according to an evolutionary algorithm until a termination condition is reached,

merging subpopulations whose processing has terminated back into the pool, and,

repeating said selecting, processing, and merging steps until an ending condition of the evolutionary algorithm is reached.

16. The media of claim 15, wherein available candidate solutions selected from the pool have relatively similar fitness values if their fitness values form a contiguous range among the fitness values of all available candidate solutions in the pool.

17. The media of claim 15, wherein each subpopulation has the same number of candidate solutions.

18. The media of claim 15, wherein each subpopulation is processed on a processor, and wherein subpopulation size is selected in dependence on the capability of the processor.

19. The media of claim 18 wherein the subpopulations are processed on two or more processors of heterogenous capabilities.

20. The media of claim 19 wherein processing a subpopulation comprises one or more generations, each generation comprising:

selecting candidate solutions from the subpopulations,

creating new candidate solutions by applying genetic operators to the selected solutions, and

evaluating the fitness of created solutions.

21. The media of claim 20 wherein the termination condition comprises processing for a fixed number of generations.

22. The media of claim 20 wherein the termination condition comprises processing until fitness of new candidate solutions ceases to increase.

23. The media of claim 15, wherein at least part of the processing of at least two subpopulations is concurrent in time.

24. The media of claim 15, wherein all steps are performed on a single processor.

25. The media of claim 15, wherein a central processor maintains the pool.

26. The media of claim 25, wherein one or more peripheral processors of heterogeneous capabilities and communicatively linked to the central processor process the subpopulations.

27. The media of claim 15, wherein the pool is maintained by communications between more than one processor.

28. The media of claim 15, wherein merging merges the incoming subpopulation into an intermediate pool of available candidate solutions ordered by fitness and transfers an outgoing subpopulation of relative similar fitness from the intermediate pool, and the software instructions further cause the one or more computers to perform an additional step of exchanging candidate solutions between the intermediate pool with the pool after a certain number of subpopulation merges.

29. A system for performing an evolutionary algorithm comprising a fitness function and an ending condition, the system comprising:

a central computer comprising a memory, and

one or more peripheral computers, each comprising a memory, the central computer and all peripheral computers being communicatively linked,

wherein the central computer memory includes a pool of available candidate solutions and one or more programs which cause this computer to perform the steps of:

selecting a subpopulation of the candidate solutions of relatively similar fitness from the pool and transmitting this subpopulation to a peripheral computer,

merging a subpopulation received from a peripheral computer back into the pool, and

repeating the steps of selecting, transmitting, and merging until an ending condition of the evolutionary algorithm is reached,

wherein each peripheral computer memory includes one or more programs which cause these computers to repetitively perform the steps of:

requesting and receiving the subpopulation from the central computer,

processing the received subpopulation according to the evolutionary algorithm until a termination condition is reached, and

returning the processed subpopulation back to the central processor.

30. The system of claim 29, wherein candidate solutions selected from the pool have relatively similar fitness values if their fitness values form a contiguous range among the fitness values of all candidate solutions in the pool.

31. The system of claim 29, wherein each subpopulation has the same number of candidate solutions.

32. The system of claim 29, wherein subpopulation size is selected in dependence on the capability of the peripheral computer.

33. The system of claim 32, wherein the peripheral computers are of heterogenous capabilities.

34. The system of claim 29 wherein processing a subpopulations comprises one or more generations, each generation comprising:

selecting candidate solutions from the subpopulations,

creating new candidate solutions by applying genetic operators to the selected solutions, and

evaluating the fitness of created solutions.

35. The system of claim 34 wherein the termination condition comprises processing for a fixed number of generations.

36. The system of claim 34 wherein the termination condition comprises processing until fitness of new candidate solutions ceases to increase.

37. The system of claim 29, wherein at least part of the processing of at least two subpopulations is concurrent in time.

38. The system of claim 29, wherein merging merges the incoming subpopulation into an intermediate pool of candidate solutions ordered by fitness and transfers an outgoing subpopulation of relative similar fitness from the intermediate pool, and the programs on the central computer cause the central computer to perform an additional step of exchanging candidate solutions between intermediate pools with the pool after a certain number of subpopulation merges.

39. The system of claim 38, wherein the central computer further performs the processing of one or more of the peripheral computers.

40. The system of claim 29, wherein the central computer further performs the processing of one or more of the peripheral computers.

41. A computer memory comprising data for implementing an evolutionary algorithm comprising a fitness function and an ending condition, the data including:

an available-pool data structure representing a central pool of available candidate solutions of the evolutionary algorithm ordered by fitness; and,

one or more local-pool data structures representing local pools of available solutions, and

control data for representing (i) transferral of candidate solutions of relatively similar fitness to a local available pool from the central available pool, (ii) processing of a local pool by a computing device according to the evolutionary algorithm until the ending condition is reached, and (iii) merging a processed local pool back into the central pool so that the central pool remains ordered by fitness.

42. The memory of claim 41 wherein the available-pool data structure resides in the physical memory of a first computing device, and wherein at least one local-pool data structure resides in the physical memory of a second computing device

43. The memory of claim 41, further comprising:

an intermediate-pool data structure representing an intermediate pool of candidate solutions, and

control data for further representing (iv) merging the local pool into the intermediate pool ordered by fitness, (v) transferral of an outgoing subpopulation of candidate solutions having relative similar fitness from the intermediate pool back into the local pool, and (vi) exchanging candidate solutions between intermediate pools with the central available pool after a certain number of subpopulation merges.

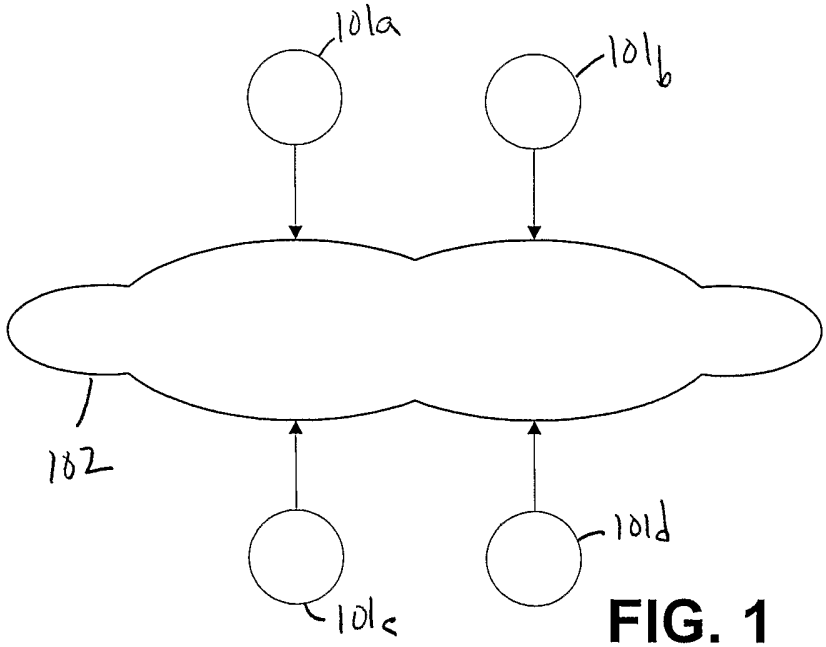


FIG. 1

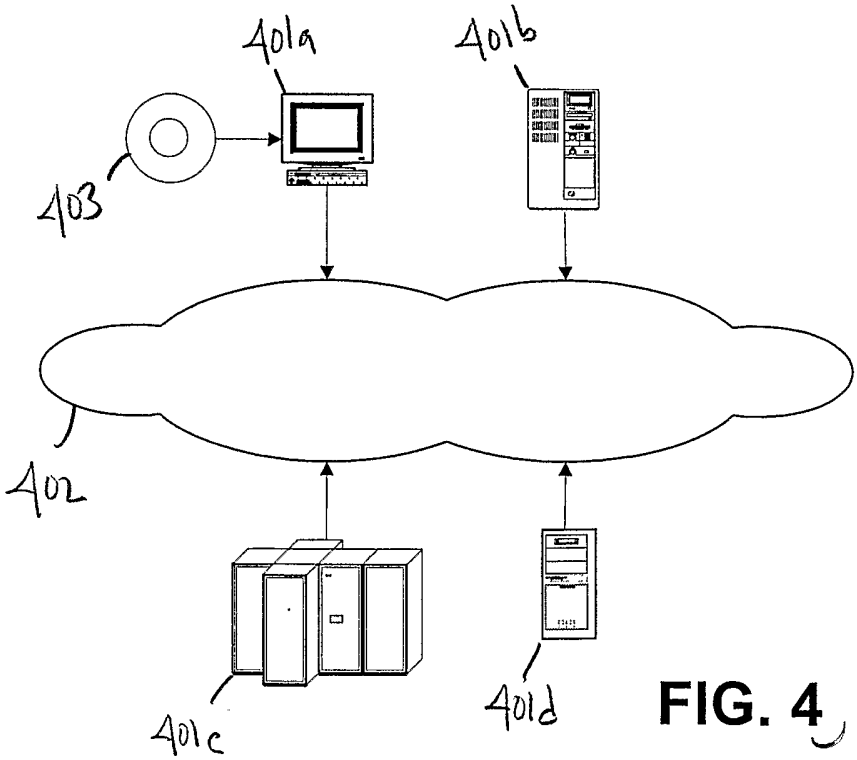


FIG. 4

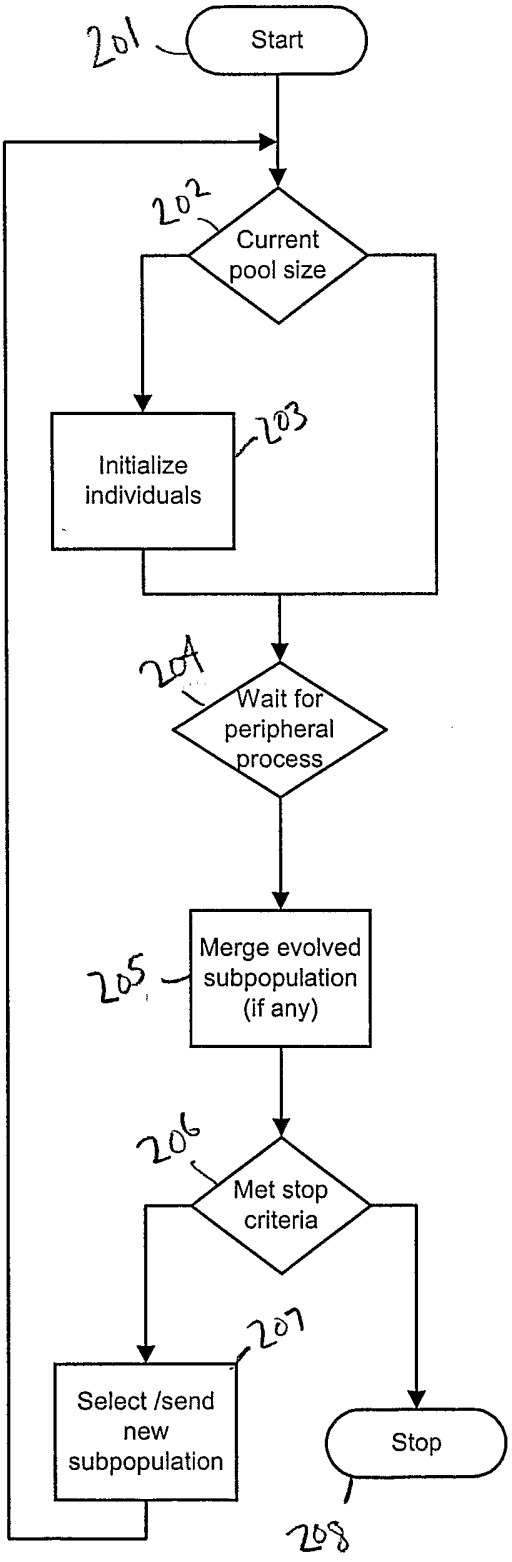

201
Start
202
Current pool size
203
Initialize individuals
204
Wait for peripheral process
205
Merge evolved subpopulation (if any)
206
Met stop criteria
207
Select /send new subpopulation
208
Stop

FIG. 2A

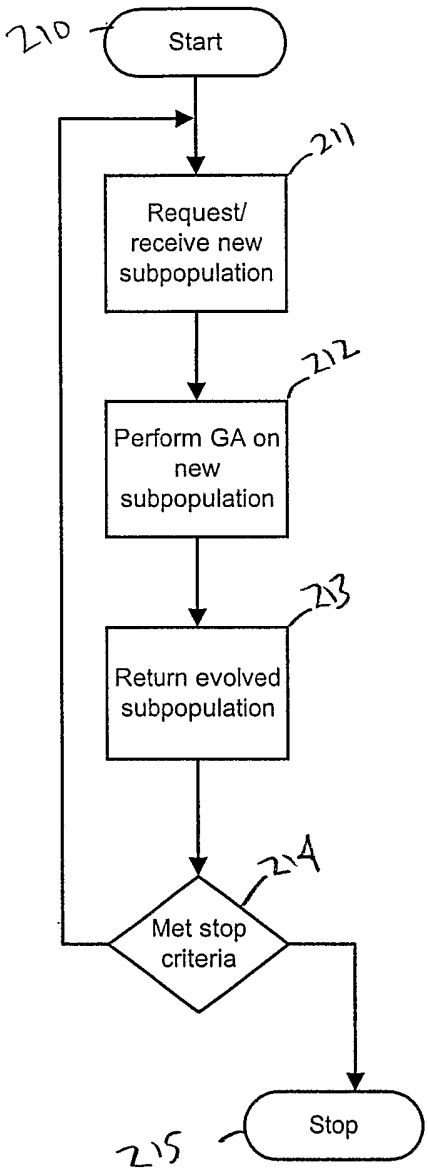

210
Start
211
Request/ receive new subpopulation
212
Perform GA on new subpopulation
213
Return evolved subpopulation
214
Met stop criteria
215
Stop

FIG. 2B

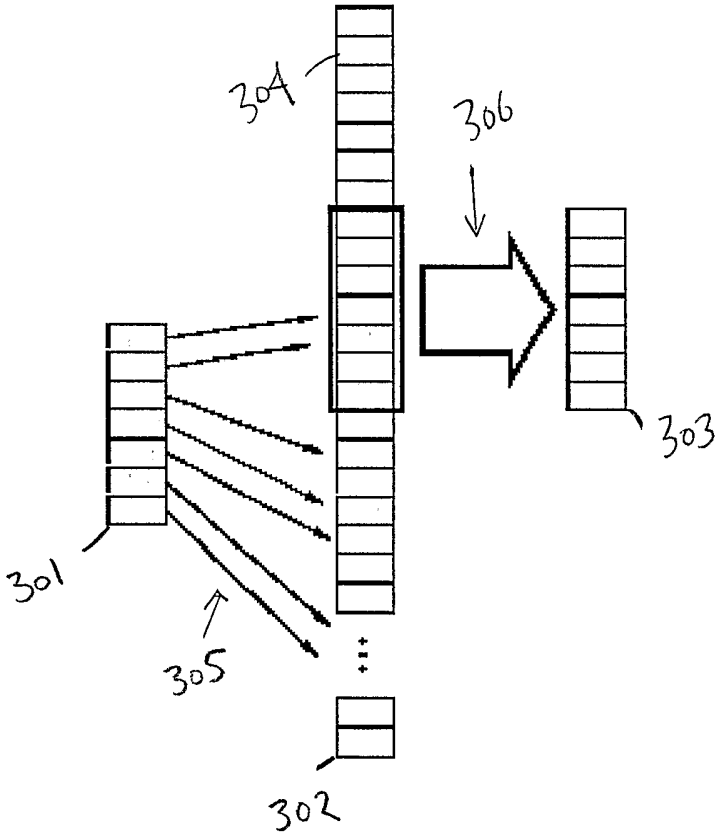


FIG. 3

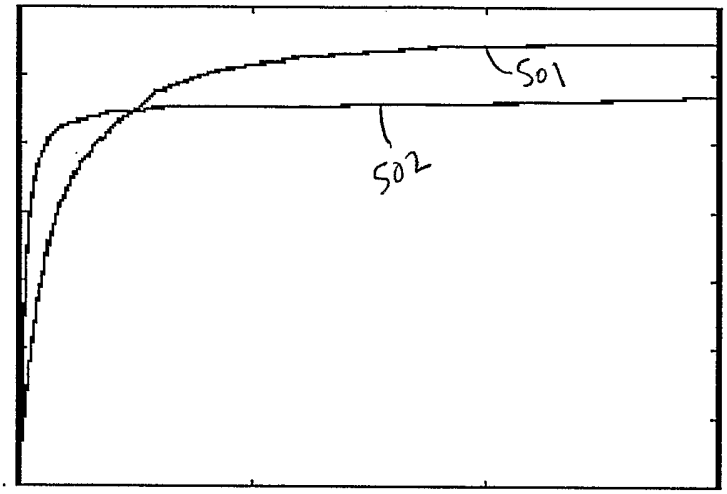


FIG. 5

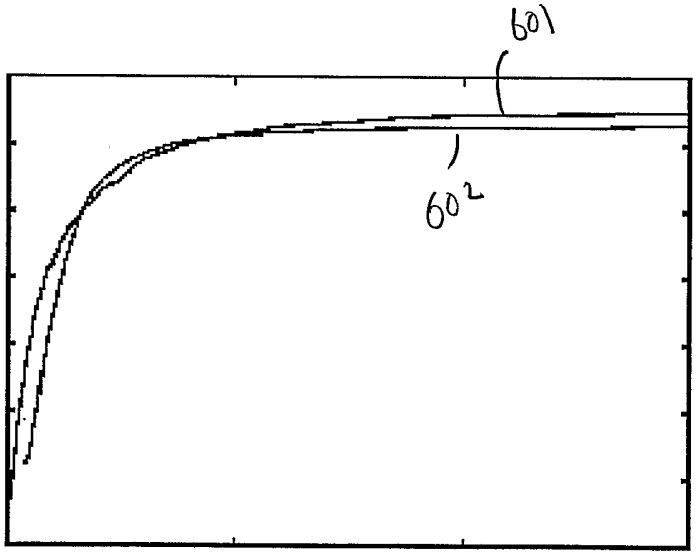


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/34571

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06N 5/02
US CL : 706/45
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)

U.S. : 706/45

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)

EAST, IEEE, ACM, INTERNET

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 2002/0156773 A1 (HILDEBRAND et al) 24 October 2002, page 1, para. 0007.	1-43
A	VALENZUELA, C.L. A Simple Evolutionary Algorithm for Multi-Objective Optimization (SEAMO), Evolutionary Computation, 2002, Vol. 1, 2002, Pages 717-722.	1-43
A	VELDHUIZEN D.A.V. et al Issues in Parallelizing Multiobjective Evolutionary Algorithms for Real World Applications, Proceedings of the 17th Symposium on Proceedings of the 2002 ACM Symposium on applied computing, March 2002, Pages 595-602.	1-43

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"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
"A"	document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier document published on or after the international filing date	"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
"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)	"&"	document member of the same patent family
"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		

Date of the actual completion of the international search 13 DECEMBER 2002	Date of mailing of the international search report 03 JAN 2003
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer Peggy Harod WILBERT L. STARKS, JR. Telephone No. (703) 308-9700

Form PCT/ISA/210 (second sheet) (July 1998)★