Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives

Zdeněk Konfršt
Czech Technical University
Department of Cybernetics, FEE
Technická 2, 160 00 Prague, Czech Republic
konfrst@zikova.cvut.cz

Abstract

This article gives a brief overview of theoretical advances, computing trends, applications and future perspectives in parallel genetic algorithms. The information is segregated into two periods before and after the year 2000 and in all chapters. The second period is more interesting and of higher importance, because it highlights recent research efforts and gives some hints about possible future trends. That is why we devote much space to the second period. As there is no such an overview of the recent period of parallel genetic algorithms, we find our investigation to be important in many aspects.

1. Introduction

Parallel genetic algorithms (PGAs) are parallel stochastic algorithms. Like sequential genetic algorithms (GAs) [27, 32, 41, 59], they are based on the natural evolutionary principle. Better individuals survive and reproduce themselves more often than the worse ones. To speed up the processing of generations of populations, we can split the population into several sub-populations and run them in the parallel way. PGAs have often been applied to various hard optimatization problems, machine learning, prediction problems and they have given interesting and good results.

1.1 Basics of Genetic Algorithms

In this part we will review some important basic terms [27, 41], that we could be able to use them later. We would like to note that PGA has a lot of things in common with a sequential or a simple GA, so we are going to start with them.

Genetic Algorithms In the beginning, there are randomly generated individuals. All those individuals create a *population*. The population in certain time is called a *generation*. According to their qualities they are chosen by operators for creation of a new generation. The quality of the population grows or decreases and give limits to some constant. Every individual is represented by its *chromosome*. Mostly chromosomes represented as a binary string. Sometimes there are more strings which are not necessarily of a binary type. The chromosome representation could be evaluated by a *fitness* function. The fitness equals to the quality of an individual and is an important pick factor for a selection process. The average fitness of a population changes gradually during the run. Operating on the population, several operators are defined. After choosing randomly a pair of individuals, *crossover* executes an exchange of the substring within the pair with some probability. There are many types of crossovers defined, but a description is beyond the scope of this report. *Mutation* is an operator for a slight change of one individual/several individuals in the population. It is random, so it is against staying in the local minimum. Low mutation parameter means low probability of mutation. *Selection* identifies the fittest individuals. The higher the fitness, the bigger the probability to become a parent in the next generation. There are different types of selection, but the basic functionality is the same.

Parallel Genetic Algorithms In PGA, there is always a selection-crossover-mutation cycle as in GAs, but you must meet new terms there. They are a deme, a migration and a topology. A *deme* is one separated population (subpopulation) in many deme populations. *Migration* means an exchange rate of individuals between the demes. It is of two types-synchronous/asynchronous. Migration has a huge impact on speed reaching the solution. It is a new process which describes how many migrants will be exchanged between the demes, when there is the right time for migration and which type of the migration schemes is useful. In parallel computation, *topology* is an important characteristics and like in the PGA. There are many types of topologies

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

between nodes/demes. Static and dynamic topologies could be used. It is worth to note that the topology brings a new dimension to GAs, because we have got several demes instead of one. Demes exchange individuals among themselves and are not anymore controlled "globally".

1.2 Parallelization and Classification

Genetic algorithms are easily parallelized algorithms. There are two kinds of possible parallelism—data parallelism and control parallelism [23]. Data parallelism involves the execution of the same procedure on multiple large data subsets at the same time.

In contrast, control parallelism involves the concurrent execution of multiple different procedures. At a given time instant each of the p processors executes a different procedure on its corresponding data set.

Naturally, data parallelism is essentially sequential; only data manipulation is parallelized and the algorithm executes one procedure in a certain period. It is the main advantage of data parallelism. The exploiting control parallelism must be carefully parallelized. The effectiveness of the control parallelism depends on several aspects of the underlying architecture of the parallel system (inter-processor topology). Data parallelism is independent of the architecture of the parallel system and simpler than the control parallelism.

In early days, most genetic parallelism was data based parallelism due to the relative simplicity. Later, some experiments with control parallelism were carried out. At present, hybrid parallelism approaches are also published to the employ advantages of both streams.

Gains from the running genetic algorithms in the parallel way are many—run time savings, speedup of finding solutions, search of greater problem space, following various diversified search paths, maximal utilization of computing machinery, increase of computational efficiency, and so on.

Many classifications, surveys, taxonomies, syntheses and overviews [15, 17, 19, 28, 44] have appeared. According to them, parallel genetic algorithms can be divided into *global*, *fine-grained*, *coarse-grained* and *hybrid* models. The classifications are also based on a walk strategy[1] (single, multiple) and on the type of (parallel) computing machinery used.

Many published papers and reports about PGAs have been applied to various theoretical problems, challenging and interesting practical applications. The publications concerning of theoretical advances are not very common as their application counterparts. We are going to start with known theory, then to move to parallel computing trends, applications and perspectives.

[1]The term of a walk strategy is derived from a simple random walk. If metaheuristic searches a problem space with one "thread", only then it uses a single walk strategy. If there are more threads, which are searching the problem space concurrently then a multiple walk strategy is used.

2 Theoretical Advances

(**Before 2000**) Bethke (1976) [9] described *global* parallel implementation of a conventional GA and a GA with a generational gap. He also showed the analysis of efficiency of using the processing capacity. He identified some bottlenecks that limit the parallel efficiency of PGAs.

Grefenstette (1981) [29] proposed four PGA types and the first three were a sort of global PGAs. They differed in accessing to (global) shared memories. The fourth type was a conventional *coarse-grained* GA. Many new questions of this PGA were raised during Grefenstette's research.

Grosso (1985) [30] proposed an implementation of a serial simulation for a concurrent formulation. Tanese's (1987) [57] and Pettey's (1987) [49] works are two of the earliest parallel implementations. The population of a GA was divided into a relatively small number of subpopulations. Each element in the architecture was assigned an entire sub-population and executed in a rather standard GA.

Cohoon (1987) [18] showed that the *punctuated equilibria* theory of the natural systems transfers to parallel implementation of evolutionary algorithms (EAs) and leads to expansion of evolutionary progress. Belding(1989) [8] implemented PGA on a hypercube parallel computer. Manderick and Spiessens (1989) [40], Gordon (1992) and Adamidis (1994) created the term of the island model parallel GA.

Very important theoretical questions were raised about *comparison of quality* solutions between a PGA and a classic GA by Starkweather, Whitley and Mathias (1999) [54]. They claimed that relatively isolated demes converge to different solutions and that migration and recombination combine partial solutions. A complete summary of the advances of the research in parallel genetic algorithms till 2000 could be found in [15, 17, 19].

(**2000 and after**) The number of papers, dissertations and books on the theory of parallel genetic algorithms have been increasing [2, 5, 15, 50, 51]. Some of them are briefly reviewed in the following.

One of the very fruitful studies was the dissertation of Cantú-Paz (2000) [15]. The dissertation brought many new principles of the rational design of fast and accurate parallel genetic algorithms. It helped many researchers to decide a configuration of the many options of topologies, migration rates, number and size of demes. The important findings were brought to light as importance of accurate population sizing for PGA, an equivalent scalability of single and multiple demes, impracticability of isolated demes, improvement quality and efficiency by migration, advantage of fully connected topologies, studies of effects of topology and optimal allocation computing resources.

Sefrioui and Périaux (2000) [51] proposed Hierarchical Genetic Algorithms (HGAs) with multi-layered hierarchical topology and multiple models for optimization problems. The architecture allowed mix of a simple and complex models, but it achieved the same quality as reached by only complex models. This solutions gave the same quality results of the nozzle reconstruction but it was three times faster when compared with the complex models.

Rivera (2001) [50] investigated how to implement parallel genetic algorithms with obtaining quality of solutions efficiently. Rivera reviewed the state-of-the-art in parallel genetic algorithms, parallelization strategies, emerging implementations and relevant results. Rivera discussed important issues regarding the scalability of parallel genetic algorithms.

Alba and Troya (2001) [2] proposed a common framework for studying PGAs. The authors analyzed the importance of synchronism in the migration step of various parallel distributed GAs. This implementation issue could affect the evaluation efforts and also provoke some differences in the search time and speedup. A set of popular evolution schemes relating to panmictic (steady-state or generational) and structured-population (cellular) GAs for the islands were used. Alba and Troya extended the existing results to structured-population GAs and demonstrated linear and even super-linear speedup when run in a cluster of workstations. In this paper, a study of several types of parallel genetic algorithms (PGAs) was published.

Alba and Troya (2002) [5] tried to bring some uniformity to the proposal, comparison, and knowledge exchanges among the traditionally opposite kinds of serial and parallel GAs. Alba and Troya comparatively analyzed the properties of steady-state, generational and cellular genetic algorithms and extended the idea to consider a distributed model consisting in the ring of the GA islands. The analyzed features were time complexity, selection pressure, schema processing rates, efficacy[2] in finding the optimum, efficiency, speedup and resistance to scalability. Besides that, they briefly discussed how the migration policy affects the search. Also, some of the search properties of cellular GAs were investigated.

Giacobini *at al.* (2003) [25] presented a theoretical study of the selection pressure in asynchronous cellular (also fine-grained) evolutionary algorithms (cEAs). The authors searched for a general model for asynchronous update of individuals in cEAs and for better models of selection intensity. The authors also characterized the update dynamics of each algorithm variant.

Xiao and Amstrong (2003) [61] proposed a new model of parallel evolutionary algorithms (EAs) called a specialized island model (SIM). The model is derived from the island model, in which an EA is divided into several subEAs that exchange individuals among themselves. In SIM, each subEA is responsible for optimizing the subset of objective functions in the initial problem. Seven scenarios of the model with a different number of subEAs, communication topology and specialization are tested and the results are compared.

[2]Efficacy means having the power to produce a desired effect. It is a measure that calculates the number of hits in finding a solution of a problem.

Gagné *at al.* (2003) [24] argued that the classic master-slave distribution model was superior to the currently more popular island-model when exploiting Beowulfs and networks of heterogenous workstations. They identified the key features of a good computing system for evolutionary computation- *transparency*, *robustness* and *adaptivity*. As far as *hard failures* caused by the network problems are concerned, they adjusted and extended the master-slave model [15] in order to considerate the possibility of those failures.

3 Trends in Computing

This section reviews some trends and fundamental issues in parallel (genetic) computing. Those are related to the designs, implementation and characteristics of parallel algorithms. Here, we will mention computer architectures, type of the computers used, network topologies and parallel languages commonly used.

3.1 Architectures

The architectures of parallel machines have emerged in the ninetities [31, 58]. Computer architectures are of Single-Instruction-Multiple-Data (SIMD) and Multiple-Instruction-Multiple-Data (MIMD) models[3]. The SIMD is known as a simpler (or a weaker) case of a parallel computer with one instruction unit and several processing units. On the other hand, the MIMD processors execute their tasks in the memory (with easy direct access), so no additional unit is necessary. As it seems that the SIMD is a weaker model than the MIMD, it can be shown that this model is equivalent up to a constant factor slowdown.

In the shared-memory machines (SIMD), all processors are able to address the whole memory space and communication between the tasks is done through read and write operations on the shared memory. The distributed-memory machines (MIMD) have their memory physically distributed among the processors. Each processor can only address its own memory, and communication among the processes executed on different processors is performed by messages passed through the communication network.

(**Before 2000**) The MIMD family evolved from shared-memory machines with a few processors (Sequence Balance, Encore Multimax) to distributed-memory machines with hundreds or more processors interconnected

[3]According to the Flynn's taxonomy.

by different topologies (hierarchical ring-the KSR[3], two-dimensional grid–the Intel Paragon, three-dimensional tore–the CRAY T3D/E, multi-stage switches the IBM-SP/SP2, Fujitsu AP1000 and so on) [8, 54, 57]. The SIMD family has been represented by massively parallel machines [7, 9, 20, 34, 40, 60] with up to 65536 (4- or 8-bit) processors such as MasPar-1 resp. 2 (MP-1, 2), the Connection Machines 1 resp. 2 (CM-1, 2).

At that time [11, 22, 55], transputer[4] networks, networks of microcomputers, have appeared as a possibility for parallel computing. They have offered fast I/O operations, huge performance and construction of a variety of networks with themselves.

(**2000 and after**) The end of the nineties has witnessed the resurgance of the shared-memory multiprocessor machines (Symetric MP or SMP). They have been equipped with two up to several hundred processors (Silicon Graphics–SGI Origin, SGI Altix 3x, Sun Microsystems-SunFire Server) [47]. But the distributed-memory machines also carry on with 64-bit RISC series (IBM pSeries, HP AlphaServer) and "clustering" series (IBM xSeries, HP RX, Dell PowerEdge, Apple Xserve) [2, 5, 15].

In recent years, the newest innovation is the connection of SMP machines via the fast local networks (Myrinet, SCI, ATM, Gigabit Ethernet) and the connection of general purpose machines via the international high speed networks. They provide scalability, fault-tolerance, excellent cost/performance ratios and so on. This has led to the emergence of clusters of computers [56] (based on Linux OS, Myrinet, off-the-shelf PCs) and has become the current hot trend in parallel computing [16, 36, 48, 52]. Cluster systems are currently among the fastest and most powerful computers according to various benchmarks[5].

3.2 OS and Topologies

(**Before and after 2000**) Parallel computers, clusters and transputer networks executed a type of UNIX[4] operating system with a sort of X-Window based interface or at least with some alike an UNIX operating system. On the market, there are also other operating systems than the UNIX/Linux mainstream such as Microsoft Windows 2003 Server, Apple Mac OS, Open VMS, but they do not compete successfully in High-Performance Computing due to various reasons (price, hardware support, scalability, maintenance).

Common underlying network topologies for parallel genetic algorithms have been multi-grids (2-D), cubes, hybercube (4-D), various meshes, toruses, pipelines, bi-directional and uni-directional rings. It is given by the type of a parallel computer and its feasibility. See more in [58].

[3]KSR stands for Kendall Square Research.

[4]http://www.inmos.com

[5]http://www.top500.org

[4]IBM SP2, pSeries–AIX; "clustering" machines–Linux clones; SGI–Irix, Sun–Solaris and INMOS Transputers–MINIX.

3.3 Libraries and Programming

(**Before and after 2000**) Basically, there are three ways to develop a parallel program. The first one is to use a parallel programming language. It is a sequential language augmented by a set of special system calls-Linda, OpenMP, HPF, Parallel C and OCCAM (both for transputer networks). The second way is that the tasks communicate by exchanging messages invoked from C/C++, FORTRAN 90, Java using communication libraries MPI (Message-Passing-Interface)[6], Express MPI, P4 (Portable Programs for Parallel Processors) or PVM (Parallel Virtual Machine) [26]. Finally, the increase of the SMP clusters is leading to the use of lightweight processes such as POSIX threads and Java threads on SMP machines. The concept of thread was extended to distributed-memory machines with tools such as Cilk, Charm++/Converse [47], Athapascan, PM2 or Java threads. More about the parallel algorithms, computation, libraries, tools, see footnote [7].

To increase efficiency in solving problems with parallel genetic algorithms, many parallel genetic libraries[8] have been implemented in the course of the "parallel genetic" years. Their characteristics are in Table 1.

#	Name	Language	Comm.	OS
1	DGENESIS	C	sockets	UNIX
2	GAlib	C++	PVM	UNIX
3	GALOPPS	C/C++	PVM	UNIX
4	PGA	C	PVM	Any
5	PGAPack	C/C++	MPI	UNIX
6	POOGAL	C++/Java	MPI	Any
7	ParadisEO	C++	MPI	UNIX

Table 1. Parallel genetic libraries and their characteristics (name, native programming language, inter-process communication and operating system).

Within the genetic domain, there are two main programming models: centralized and distributed. The centralized model, also called master-slave or client-server, handles data by one processor (master) or stored in a shared memory. The distributed model is characterized by a lack of global data, either shared or centralized. Information is shared or made global by the exchange of messages among

[6]http://www.mpi.nd.edu/lam

[7]http://wotug.ukc.ac.uk/parallel/

[8]http://www.aic.nrl.navy.mil/galist/src/

the processors. With the advent of clusters of SMP machines, many research works implemented a hybrid model- a centralized model within each SMP machine, but running under a distributed model within machines in the cluster.

4 Applications

Applications of PGAs (GAs) are regularly wide and range from Numerical Mathematics and Graph Theory (numerical function optimatizations, graph bipartity, graph partitioning problem, scheduling problems, mission routing problems), through Computer Science (searching for weights of neural networks, optimization of server load or database queries), Finance and Economics (financial balancing problems, transport problems, modelling systems, predictions of time series) to Technology and Engineering (optimization of VLSI circuits, optimization of car wheels, optimization in Material Engineering).

(**Before 2000**) A very good overview of applications of parallel genetic algorithms was published in [15, 17, 19].

(**2000 and after**) Solano *at al.* (2000) [53] worked on an approach to implement, in real-time, a parametric spectral estimator method using genetic algorithms (GAs) and to find the optimum set of parameters for the adaptive filter that minimises the error function for Doppler ultrasound signals. The primary aim was to reduce the computational complexity of the conventional algorithm by using the simplicity associated to GAs and exploiting its parallel characteristics.

Oyama *at al.* (2000) [46] applied PGA to a practical three-dimensional shape optimization for aerodynamic design of a transonic aircraft wing. The authors called the algorithm-ARGA (Real-coded Adaptive Range Genetic Algorithm), which had both binary and real value representations. Aerodynamic optimization gave a very enhanced wing design, which has shown feasibility of the parallel genetic approach.

Moser and Murty (2000) [42] applied a scalable distributed genetic algorithm to a very large-scale feature selection. The domain application was a classification system for Optical characters, namely hand-written digits. The algorithm was capable of reduction of the problem complexity significantly and scale very well according to very large-scale problems.

Alba and Troya (2000) [1] considered both panmictic and structured-population algorithms as two reproductive loop types executed in the islands of a parallel distributed GA. Their aim was to extend the existing studies from more conventional sequential islands to other kinds of evolution. A key issue in such a coarse grain PGA was the migration policy, since it governs the exchange of individuals among the islands. They also investigated the influence of migration frequency and migrant selection in a ring of islands running either steady-state, generational, or cellular GAs with different problem types, namely easy, deceptive, multimodal, NP-Complete, and epistatic search landscapes in order to provide a wide spectrum of problem difficulties to support the results.

Chalermwat *at al.* (2001) [16] presented 2-phase sequential and coarse-grained parallel image registration algorithms using GAs as optimization mechanism. In its first phase, the algorithm found a small set of good solutions using low-resolution versions of the images. Based on these candidate low-resolution solutions, the algorithm used the full resolution image data to refine the final registration results in the second phase. Experimental results were presented and revealed that algorithms had yielded very accurate registration results for LandSat Thematic Mapper images[9], and the parallel algorithm scaled quite well on the Beowulf parallel cluster.

Bevilacqua *at al.* (2001) [10] investigated the improvement obtained by applying a distributed genetic algorithm to a problem of parameter optimization in the medical images analysis. The authors set a method for the detection of clustered microcalcifications in digital mammograms, based on statistics and multi-resolution analysis by means of wavelet transform. A distributed genetic algorithm supervised the process of fluctuation of detection parameters to improve detection results.

Olague (2001) [45] implemented a system for placing cameras in order to satisfy a set of interrelated and competing constrains for three-dimensional objects. The system provided the attitude of each camera in the network, taking into account the imaging geometry, visibility, convergence angle and workspace constraints.

Alba and Troya (2002) [4] implemented a distributed PGA in Java that run at the same time on different machines linked by different kinds of communication networks. This algorithm benefited from the computational resources offered by modern LANs and by the Internet. They analyzed the way in which such heterogeneous systems affect the genetic search for two problems.

Fan *at al.* (2002) [21] used PGAs for mutual information-based registration of medical image data of different modalities and multiple times from computer tomography and magnetic-resonance imaging sources as a part of medical image analysis. The presented genetic strategy produced extremely robust results with super-linear speedup in the subpopulation manner.

Pelikan *at al.* (2002) [47] described an implementation of a fine-grained parallel genetic algorithm. The fine-grained genetic algorithm implemented in Charm++, a message-driven parallel language based on C++, was described. The implementation was fully asynchronous and distributed. Thus, it scaled well, even for a very large num-

[9]http://www.landsat.org/

ber of processors. The performance results for up to 64 processors on an Origin2000 verified scalability hypothesis. The implementation allowed solutions represented by binary strings and decision graphs with Boolean, real-valued, and integer attributes. Fitness functions included a simple linear problem for binary strings and classification of data sets which are dynamically loaded from a specified data file. Any of the components could easily be extended by deriving a new class for the representation and fitness functions.

Jelasity *at al.* (2002) [33] proposed a tool for automatic learning of algorithm components based on distributed evolutionary algorithms for problem classes. The tool was called DRM (distributed resource machine) as a part of the DREAM project [6]. DRM, a supporter of the conceptual framework of the multi-agent system implemented in Java, was capable of running distributed experiments on the Internet, ideally suited for algorithm learning. Tool tests were run against a subset sum problem.

Arenas *at al.* (2002) [6] released DREAM[10] (Distributed Resource Evolutionary Algorithm Machine) framework for the automatic distribution of evolutionary algorithm processing through a virtual machine built from a large number of individual computers on the Internet. The framework contained five user entry points to access it. The highest level used evolutionary algorithms within the graphical displays. The lowest level of the framework was a Peer to Peer mobile agent system that could distribute a class of evolutionary algorithm processes.

Pereira (2003) [48] explored the use of the Island Genetic Algorithm (IGA), a coarse-grained parallel GA model, comparing its performance to that obtained by the application of a traditional non-parallel GA. The optimization problem consisted of adjusting several reactor cell parameters, such as dimensions, enrichment and materials, in order to minimize the average peak-factor in a three-enrichment zone reactor, considering the restrictions on the average thermal flux, criticality and sub-moderation. The IGA implementation was run as a distributed application on a conventional local area network (LAN), avoiding the use of expensive parallel computers or architectures. After exhaustive experiments, the IGA provided gains not only in terms of computational time, but also in the optimization outcome.

Kwon and Moon (2003) [38] proposed a neuro-genetic daily stock prediction model. Traditional indicators of stock prediction are utilized to produce useful input features of neural networks. The genetic algorithm optimizes the neural networks under a 2D encoding and crossover. A parallel genetic algorithm was used on a Linux cluster. A notable improvement on the average buy-and-hold strategy was observed.

[10]http://www.dcs.napier.ac.uk/benp/dream/dream.htm

5 Discussion

In the theory, we can observe many parallel genetic models to understand their behaviour, speed-up of solution generation and use an appropriate type of PGAs for a special problem. As has been reviewed, there are models like principles of a rational design, accurate prediction models, hierarchical genetic algorithms, quality of solutions and others. As far as the parallel genetic computing platform is concerned, Linux clusters with Myrinet (or Gigabit Ethernet) or SMP clusters are the main stream these days. Developing tools are Java, C/C++ and MPI and also Java threads.

Applications have diversified in many new application areas. Based on the article, the applications were from the following areas: optimization of ultrasound signals, optimization of aerodynamic design of an ultrasonic aircraft wing, optical large-scale feature selection, image registration, medical images analysis, a system for placing cames, using computing sources of the Internet-grid computing, automatic learning algorithms, optimizations in nuclear engineering, stock prediction and software development of parallel evolutionary libraries.

6 Perspectives

In this section, we forecast the directions of research of parallel genetic algorithms till the year **2005**. We expect many more theories of PGAs based on various approaches. Only a brief list of possible theories is given: representation theories, operator theories, convergence theory, theory of structured algorithms, theory of fitness landscape, unification theory, working models theories and speciation theories and niches.

In our view, approximations and approximation theories based on a population size, problem difficulty, topology, time bounding, parallel computer parameters are among the most important and useful ones.

In parallel computing, we expect more development and use of powerful Linux clusters, new parallel object-oriented languages [12, 47] and Java language with OO design and parallel threads, support of distributive computing and support of various other standards and technologies(XML-processing, J2EE, RMI).

Also, new parallel programming libraries, like PVM and MPI have been, may appear. One of them is OpenMP[11]. It is a set of compiler directives and library routines to express shared memory parallelism. The majority of OpenMP is a set of compiler directives that says to a sequential program which parts will be run concurrently and where to put synchronization points. OpenMP gives a chance to parallelize the existing sequential software.

[11]http://www.openmp.org

The emergence of evolutionary algorithm libraries and frameworks is also expected similar to DREAM [6, 33], ParadisEO [13, 14] or further expansion of the existing ones. Generally, the frameworks will be tightly connected with the object-oriented paradigm, C++/Java programming languages, a wide distributed environment-Internet and easy web access and use.

In the application area, we expect many more applications of parallel genetic algorithms in the domains of knowledge discovery and data mining as shown in [23, 39]. These days, there is a huge amount of data stored in real-world databases and the amount grows very fast. The need is to discover the knowledge hidden in these databases by intelligent automatic methods. If such important knowledge discovery has been made, the decision making process will be improved. The applicability in the business world is enormous with increasing potential in the future.

The research fields, which deal with image information like computer recognition, computer vision and image processing [10, 16, 21, 42, 45], have been extremely promising targets for parallel genetic algorithms. Relevant fields like signals filtering and processing [53, 60] will also be more frequently targeted.

7 Conclusion

The article is a survey of past and recent developments in parallel genetic algorithms. The main focus has been developing since the year 2000. The relevant issues connected with parallel genetic algorithms were highlighted. The survey hinted several views, whose range from new genetic theories over parallel computing and wide and various ranges of real-world applications to future developments, challenges and perspectives for parallel (genetic) metaheuristcs.

References

[1] E. Alba, J. M. Troya. Influence of the Migration Policy in Parallel Distributed GAs with Structured and Panmictic Populations. *Applied Intelligence*, 12(3):163-181, 2000.

[2] E. Alba, J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Generation Computer Systems*, 17(4):451-465, 2001.

[3] E. Alba. Parallel evolutionary algorithms can achieve superlinear performance. *Information Processing Letters*, 82(1):7-13, 2002.

[4] E. Alba, A. J. Nebro, J. M. Troya. Heterogeneous Computing and Parallel Genetic Algorithms. *Journal of Parallel and Distributed Computing*, 62(9):1362-1385, 2002.

[5] E. Alba, J. M. Troya. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing* , 12(2):91-114, 2002.

[6] M. Arenas, P. Collet, A. Eiben, M. Jelasity, J. Merelo, B. Paechter, M. Preuß, M. Schoenauer. A Framework for Distributed Evolutionary Algorithms. *Parallel Problem Solving from Nature VII*, Granada, Spain, 665-675, 2002.

[7] S. Baluja. Structure and Performance of Fine-Grained Parallelism in Genetic Search. *Fifth Int. Conf. on Genetic Algorithms*, San Mateo, CA, 155-162, 1993.

[8] T. C. Belding. The distributed genetic algorithm revisited. *Proceedings of the Sixth International Conference on Genetic Algorithms*, San Francisco, CA, 114-121, 1995.

[9] A. D. Bethke. *Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Efficiency of use of processing capacity*. Tech. Rep. No. 197, Logic of Computers Group, University of Michigan, 1976.

[10] A. Bevilacqua, R. Campanini, N. Lanconelli. A Distributed Genetic Algorithm for Parameters Optimization to Detect Microcalcifications in Digital Mammograms. *Applications of Evolutionary Computing EvoWorkshops:EVOIASP*, Como, Italy, 278-287, 2001.

[11] R. Bianchini, C. M. Brown. Parallel genetic algorithms on distributed-memory architectures. *Transputer Research and Applications*, IOS Press, Amsterdam, 6:67-82, 1993.

[12] M. Bubak, K. Sowa. Object-oriented implementation of parallel genetic algorithms. *High Performance Cluster Computing: Programming and Applications*, 2:331-349, 1999.

[13] S. Cahon, E-G. Talbi, N. Melab. ParadisEO: A Framework for Parallel and Distributed Metaheuristics. *IPDPS'03*, Nice, France, 144a, 2003.

[14] S. Cahon, E-G. Talbi, N. Melab. ParadisEO: A Framework for the Flexible Design of Parallel and Distributed Hybrid Metaheuristics. *Issue on Parallel Computing*, Elsevier Science, 2003.

[15] E. Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, 162, 2000.

[16] P. Chalermwat, T. El-Ghazawi, J. LeMoigne. 2-phase GA-based image registration on parallel clusters. *Future Generation Computer Systems*, 17(4):467-476, 2001.

[17] T. G. Crainic, M. Toulouse. *Parallel Metaheuristics*. Centre de recherche sur les transports Université de Montréal, Canada, 53, 1997.

[18] J. P. Cohoon, S. U. Hedge, W. N. Martin . Punctuated equilibria: a parallel genetic algorithm. *Proc. Second Int. Conf. on Genetic Algorithms*, Pittsburg, PA, 148-154, 1987.

[19] V. Cung, L. S. Martins, C. C. Ribeiro, C. Roucairol. *Strategies for the Parallel Implementation of Metaheuristics*. Laboratoire PRiSM-CNRS, Université de Versailles, France, 33, 2001.

[20] R. E. Dorsey. Non-linear Optimization on a Parallel Intel i860 RISC Based Architecture. *Computational Economics*, 10(3):279-294, 1997.

[21] Y. Fan, T. Jiang, D. Evans. Medical Image Registration Using Parallel Genetic Algorithms. *Applications of Evolutionary Computing EvoWorkshops: EVOIASP Talks*, Kinsale, Ireland, 304-314, 2002.

[22] T. C. Fogarty, R. Huang. Implementing the genetic algoritm on transputer based parallel processing systems. *Parallel Problem Solving from Nature*, 145-149, 1991.

[23] A. A. Freitas. *Datamining and knowledge discovery with evolutionary algorithms*. Springer, Berlin, 264, 2002.

[24] Ch. Gagné, M. Parizeau, M. Dubreuil. The Master-Slave Architecture for Evolutionary Computations Revisited. *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, IL, 2:1578-1579, 2003.

[25] M. Giacobini, E. Alba, M. Tomassini. Selection Intensity in Asynchronous Celluar Evolutionary Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, IL, 2:955-966, 2003.

[26] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam. *PVM 3 User's Guide and Reference Manual*. Oak Ridge National Laboratory, Tennessee, 279, 1994.

[27] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 412, 1989.

[28] A. Grajdeanu. *Parallel Models for Evolutionary Algorithms*. ECLab, George Mason University, 38, 2003.

[29] J. J. Grefenstette. *Parallel adaptive algorithms for function optimization*. Report No. CS-81-19, Vanderbilt University, TN, 1981.

[30] P. Grosso. *Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model*. PhD thesis, Department of Computer and Communications Sciences, University of Michigan, 1985.

[31] J. Hlavička. *Computer architecture*. Czech Technical Publishing House, 206, 1997.

[32] J. H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[33] M. Jelasity, M. Preuß, A. Eiben. Operator Learning for a Problem Class in a Distributed Peer-to-Peer Environment. *Parallel Problem Solving from Nature VII*, Granada, Spain, 172-183, 2002.

[34] U. Kohlmorgen, H. Schmeck, K. Haase. Experiences with fine-grained parallel genetic algorithms. *Annals of Operations Research*, 90:203-219, 1999.

[35] Z. Konfršt, J. Lažanský. The Population Sizing Problem in (P)GAs: Experiments. *Intelligent Technologies: Theory and Applications*, IOS Press, Amsterdam, The Netherlands, 146-151, 2002.

[36] Z. Konfršt, J. Lažanský. Extended Issues of PGAs based on One Population. *Neuro Fuzzy Technologies '2002*, Havana, Cuba, 71-78, 2002.

[37] Y. Kwoka, I. Ahmada. Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors Using a Parallel Genetic Algorithm. *Journal of Parallel and Distributed Computing*, 47(1):58-77, 1997.

[38] Y. Kwon, B. Moon. Daily Stock Prediction Using Neuro-genetic Hybrids. *Proceedings of the Genetic and Evolutionary Computation Conference*, Chicago, IL, 2:2203-2214, 2003.

[39] Q. Liu, S. M. Bridges, I. Banicescu. *Parallel Genetic Algorithms for Tuning Fuzzy Data Mining System*. Tech. Report, Mississippi State University, 6, 2001.

[40] B. Manderick, P. Spiessens. Fine-Grained Parrallel Genetic Algorithms. *Third International Conference on Genetic Algorithms*, San Mateo, CA, 428-433, 1989.

[41] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlang, 252, 1992.

[42] A. Moser, N. Murty. On the Scalability of Genetic Algorithms to Very Large-Scale Feature Selection. *Real-World Applications of Evolutionary Computing: EvoWorkshops*, Edinburgh, Scotland, 77-86, 2000.

[43] H. Mühlenbein, M. Schomisch, J. Born. The parallel genetic algorithm as function optimizer. *Proc. of the Fourth Intl. Conf. on Genetic Algorithms*, San Mateo, CA, 271-278, 1991.

[44] M. Nowostawski, R. Poli. Parallel Genetic Algorithm Taxonomy. *Proceedings of 3rd International Conference on Knowledge-based Intelligent Information Engineering Systems*, Adelaide, Australia, 1999.

[45] G. Olague. Autonomous Photogrammetric Network design Using Genetic Algorithms. *Applications of Evolutionary Computing EvoWorkshops*, Como, Italy, 353-364, 2001.

[46] A. Oyama, S. Obayashi, T. Nakamura. Real-Coded Adaptive Range Genetic Algorithm Applied to Transonic Wing Optimization. *Parallel Problem Solving from Nature VI.*, Paris, France, 712-721, 2000.

[47] M. Pelikan, P. Parthasarathy, A. Ramraj. Fine-grained Parallel Genetic Algorithms in Charm++. *ACM Crossroads Magazine: Parallel Computing*, 8(3), 2002.

[48] C. Pereira, C. Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. *Annals of Nuclear Energy*, 30(5):555-565, 2003.

[49] C. Pettey, M. Lenze, J. J. Greffenstette. A Parallel Genetic Algorithm. *Proc. of 2nd ICGA*, San Mateo, CA, 155-161, 1987.

[50] W. Rivera. Scalable Parallel Genetic Algorithms. *Artificial Intelligence Review*, 16(2):153-168, 2001.

[51] M. Sefrioui, J. Périaux. A Hierarchical Genetic Algorithm Using Multiple Models for Optimization. *Parallel Problem Solving from Nature VI.*, Paris, France, 879-888, 2000.

[52] G. A. Sena, D. Megherbi, G. Isern. Implementation of a parallel genetic algorithm on a cluster of workstations: Traveling salesman problem, a case study. *Future Generation Computer Systems*, 17:477-488, 2001.

[53] J. Solano González, K. Rodríguez Vázquez, D. F. García Nocetti. Model-based spectral estimation of Doppler signals using parallel genetic algorithms. *Artificial Intelligence in Medicine*, 19(1):75-89, 2000.

[54] T. Starkweather, D. Whitley, K. Mathias. Optimization using distributed genetic algorithms. *Parallel Problem Solving from Nature*, Berlin, Germany, 176-185, 1991.

[55] M. Schwehm. Implementation of genetic algorithms on various interconnections networks. *Parallel Computing and Transputer Applications*, 195-203, 1992.

[56] T. Sterling, J. Salmon, J. D. Becker, F. D. Savarese. *How to Build a Beowulf. A Guide to the Implemnetation and Application of PC Clusters*. The MIT Press, 239, 1999.

[57] R. Tanese. *Distributed genetic algorithms for function optimization*. Unpublished doctoral thesis, University of Michigan, Ann Arbor, 1989.

[58] P. Tvrdík. *Parallel systems and algorithms*. Czech Technical Publishing House, 167, 1994.

[59] M. Vose. *The Simple Genetic Algorithm: Foundation and Theory*. MIT Press, 251, 1999.

[60] L. Yang, M. Misra. Parallel Algorithms for Multi-Dimensional Wavelet Transforms. *The Journal of Supercomputing*, 12(1-2):99-118, 1998.

[61] N. Xiao, M. Amstrong. A Specialized Island Model and Its Application in Multiobjective Optimization. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:1530-1540, Chicago, IL, 2003.