# Parallel Genetic Algorithms: A Useful Survey

TOMOHIRO HARADA*, Ritsumeikan University, Japan
ENRIQUE ALBA, Universidad de Malága, Spain

In this article we encompass an analysis of the recent advances in parallel genetic algorithms (PGAs). We have selected these algorithms because of the deep interest in many research fields for techniques that can face complex applications where running times and other computational resources are greedily consumed by present solvers, and PGAs act then as efficient procedures that fully use modern computational platforms at the same time that allow the resolution of cutting edge open problems. We have faced this survey on PGAs with the aim of helping newcomers or busy researchers who want to have a wide vision on the field. Then, we discuss the most well-known models and their implementations from a recent (last five years) and useful point of view: we discuss on highly cited articles, keywords, the venues where they can be found, a very comprehensive (and new) taxonomy covering different research domains involved in PGAs, and a set of recent applications. We also introduce a new vision on open challenges, and try to give hints that guide practitioners and especialized researchers. Our conclusion is that there are many advantages in using these techniques, and lots of potential interactions to other evolutionary algorithms, as well as we contribute to create a body of knowledge in PGAs by summarizing them in a structured way, so that the reader can find this article useful for practical research, graduate teaching, and as a pedagogical guide to this exciting domain.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Mathematics of computing → Evolutionary algorithms**; • **Theory of computation → Evolutionary algorithms**; • **Computing methodologies → Parallel algorithms**; **Genetic algorithms**;

Additional Key Words and Phrases: Parallelism, Genetic Algorithms, Complex Problem Solving, Time Consuming Applications, New Areas for Search, Optimization, and Learning, Last Five Years, Artificial Intelligence

**ACM Reference Format:**
Tomohiro Harada and Enrique Alba. 2018. Parallel Genetic Algorithms: A Useful Survey. *ACM Comput. Surv.* 9, 4, Article 39 (November 2018), 35 pages. https://doi.org/0000001.0000001

## 1 INTRODUCTION

Let us start this article by answering the most important obvious question: why should we bother in reading on Parallel Genetic Algorithms (PGAs)? The answer is not that obvious as one would think ("because we can solve complex problems in a more efficient manner") because it goes to the root of world research, to engineering, and to the importance of optimization in all this (as well as search and machine learning). If we focus on optimization, we soon will notice that most disciplines and works on apparently diverse fields finally rely in optimizing a complex system: from building new drugs [44, 56, 83] (optimize the energy estate of chemicals, like their angles), deploying better 5Gs networks [51] (optimized covered area, the packet protocol parameters and the number of potential users), investing in the stock market [22, 112] (maximize revenues of a complex portfolio), or helping

*This is the corresponding author

Authors' addresses: Tomohiro Harada, Ritsumeikan University, 1-1-1 Nojihigashi, Kusatsu, Shiga, 5258577, Japan, harada@ci.ritsumei.ac.jp; Enrique Alba, Universidad de Malága, Malága, Spain, eat@lcc.uma.es.


0360-0300/2018/11-ART39 $15.00
https://doi.org/0000001.0000001

citizens to move in a city [110, 111, 118, 121] (minimize pollution, fuel consumption, travel time, maximize comfort and driver's experience), just to name a few. Filling your fridge with healthy food at low cost, combat a fire, program your computer with a minimum number of bugs, and thousands of other applications are now possible by using GAs [6].

These *metaheuristics* (also known as *bio-inspired techniques*) [24, 80, 115] can address applications where the target goal function is non-derivable, non-continuous, ill-defined, uses arbitrary mixed types of variables, has many competing goals, or even does not have any analytical expression at all. This wide applicability made GAs to penetrate most domains of research, development, and innovation [7]. The massive benefits of these techniques took them to multidisciplinary applications, and starred lots of success stories in their short-term (thirty years) existence. And, paired to this impressive presentation card, a main problem always arrives to practitioners and researchers: studies and products based in GAs need long running times as soon as they start being useful. This long running times come to scene for many reasons: (a) a large dimensionality of the problem, (b) a need to use customized operators for the application, (c) a complex target function that takes several minutes of computation (d) a need of dealing with large datasets in the algorithm, (e) a high number of non-linear restrictions...and still for some other reasons not shown here. When researchers arrive to this *wall*, parallelism is one way to jump over it for an improved performance.

But long running times are just one way to arrive to PGAs. In fact, it was quite a while since researchers have shown that PGAs are not just faster algorithms. They are much more than this, they are new models of research that can profit from parallel platforms like clusters, multiprocessors, GPUs, and others [4, 117]. Indeed, an additional source for speed up (time gains) is the potential creation of new techniques that search, optimize and learn in a numerically faster way, by saving function calls (the usual way of measuring numerical efficiency). Thus, much can be gained by e.g. running a distributed algorithm on a network [15, 54, 127], or a cellular model on a GPU [125]. This separation between model and implementation is worth knowing, and we will organize one of our sections according to this.

In facing this survey, we had to make decisions. First, we decided not to go for evolutionary algorithms (EAs) in general, since the topic of PGAs will fade away. In fact, GAs are maybe the most popular class of EAs, and most things done on them can be reproduced in other EAs and population based techniques, so it seems worthy to focus-to-tell, instead of making a too abstract survey. Another decision was on actually going beyond normal surveys that only revise papers and do not offer too much added value. We intentionally tried to offer some contributions to make this a proactive and useful survey, in particular:

- We offer a chronological survey of articles and results in PGAs, so that the reader grasps how the trends are evolving at world level.
- We make an initial discussion on PGA models to make the survey pedagogical and self-contained (to an extent). This also helps to make better proposals for those interested in building models and later implement them in parallel platforms.
- We summarize the most cited papers in the last five years (fresh vision) and point out the most important venues (both, journals and conferences) so that researchers know where to find new papers in the future and where to head if interested in publishing their results on PGAs.
- We give a vision on the most important keywords in the domain, to offer a comparative idea of the importance of topics at this moment.
- We analyze authors, and how the domain is attracting attention of both, new and experienced researchers.
- We analyze the higher impact papers (in number of cites) in a semantic consistent way, as we also delve into their implementation, platforms, programming languages, and comments on their contents relevant for other practitioners.

- We analyze PGAs from the point of view of the applications where they are used, thus showing where they are successful (by looking at what we include) and where are some opportunities for new applications (by thinking in what it is not found in recent references).
- We dare to list several future research lines where appealing challenges can be found, so as to help in defining new PhD theses and master works involving PGAs.

This survey is structured in several sections. The next one will be devoted to a methodological description of PGAs. Then, Section 3 will develop on important and recent venues for finding/publishing results with PGAs. Later, in Section 4, we address the details found in recent articles on PGAs. Finally, in Section 5 we describe our vision on future challenges, and end the paper with several conclusions.

## 2 PARALLEL GENETIC ALGORITHM

This section gives a brief overview of GA and PGA, and shows a new and comprehensive taxonomy of PGA. Then, we introduce implementations of PGA, which are usually employed in conventional PGA researches.

### 2.1 Overview

A genetic algorithm (GA) is a powerful metaheuristic inspired by Darwin's theory of nature evolution. Algorithm 1 shows a pseudo-code of a standard GA. A GA is a population-based algorithm, in which many tentative solutions are managed. Firstly, an initial population $P_0$ is constructed based on an initialization method (`initialization`). An initial population is (often) generated in a random uniform manner, while some heuristics such as Latin hypercube sampling method [79] or problem specific approaches are employed to give better starting points for the search. Tentative solutions in the population are evaluated depending on the fitness function (`evaluation`), that assesses how much a tentative solution is fitted to a target problem. After that, the main procedure of the GA is executed. In the main procedure, new tentative solutions $P'_t$ are generated through variation operators (`variation`) such as crossover and mutation, and their fitness is evaluated. Then, a next population $P_{t+1}$ is constructed from the current population $P_t$ and the newly generated tentative solutions $P'_t$ (`replacement`). These procedures are repeated until a pre-defined termination criterion is satisfied, such as the quality of the desired solution, the maximum number of evaluations, or the computing time.

A GA is well suited to apply parallelization because of its population based approach, where all solution candidates can be dealt with in parallel. When extending a sequential GA with parallel techniques, we have to consider several characteristics of the built PGA. A new and comprehensive taxonomy of PGA is shown in Fig. 1, because these considerations and characteristics are quite a few and need some structure. This taxonomy consists of six main and indispensable components of PGA: `Implementation`, `Hardware`, `Software`, `API`, `Applications`,

**ALGORITHM 1:** Pseudo-code of a genetic algorithm

```
t ← 0;
initialization(P_0);
evaluation(P_0);
while Termination criterion do
    P'_t ← variation(P_t);                                  // Change existing tentative solutions
    evaluate(P'_t);                  // Compute the relative quality of every solution in the population
    P_{t+1} ← replacement(P_t, P'_t);
    // Merge the auxiliary and previous populations and select a new better one
    t ← t + 1;
end
```

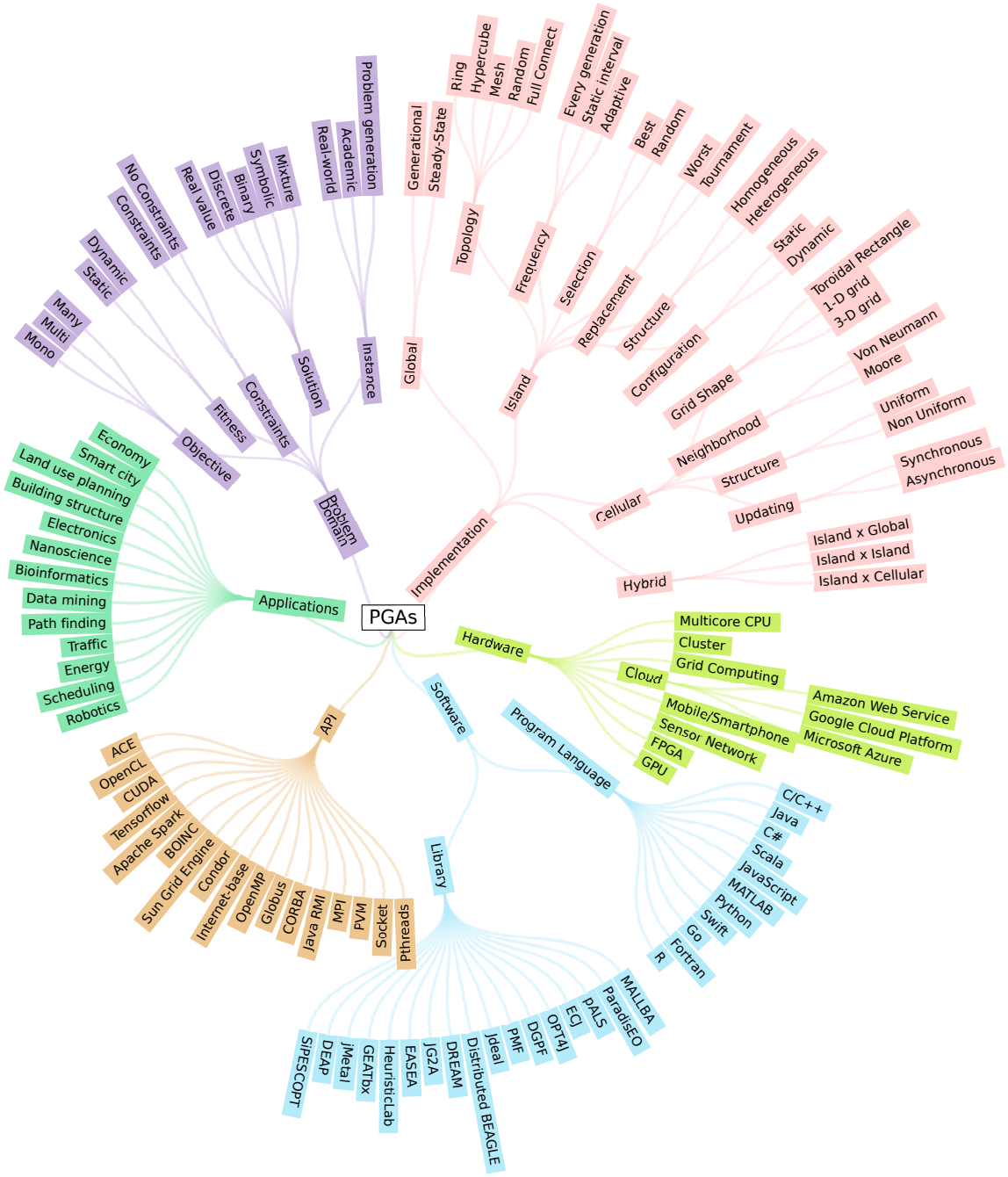


Fig. 1. A new comprehensive taxonomy of Parallel Genetic Algorithms

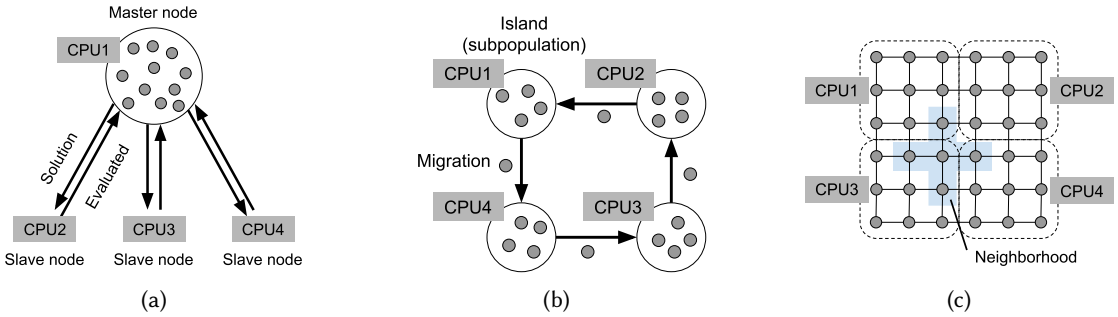


Fig. 2. Three usual PGA models: (a) Global parallelization, (b) Parallel island model, (c) Parallel cellular model

and `Problem domain`, which are indicated at the first layer. Each component is represented with different colors: `Implementation` in pink, `Hardware` in blue, `Software` in green, `API` in orange, `Applications` in purple, and `Problem domain` in cyan. Deeper layers (sub-taxonomies) of each component present further details. Such classification is also presented in previous PGA surveys. In the survey of Alba et al. in 2002 [13], authors focus on PGA models, hardware architectures, and APIs used to implement PGAs. The survey of Alba et al. in 2013 [10] focuses on PGA models, their applications, hardware architectures, and software including APIs, although it discusses not only PGA, but also parallel metaheuristics, including trajectory-based approach and population-based ones. The survey of Talbi in 2015 [116], which mainly focuses on parallel evolutionary combinational optimization, divides parallel designs into three levels: algorithmic-level, iteration-level, and solution-level and discusses on hardware architecture. However, in his survey, software, applications, and problem domains are not taken into account. Our proposed taxonomy, in contrast to them, comprehensively incorporates these elements necessary for the realization of PGA and represents them all together in a single figure.

`Implementation` indicates how to implement a GA when going for any parallel methodology. This topic will be discussed soon in the next subsection. `Hardware` is an important factor of PGA because it is the source of many important time gains, and influences important operations like population management or communication model. The most popular hardware actually employed in PGA literature is discussed in Section 4.2. `Software` and `API` are also important factors of PGA. In this taxonomy, `Software` consists of `Library` and `Program Language`. Several libraries for GA are presented by researchers, while the choice of program language highly relates to not only implementation of PGAs, but also to their computing speed. API supports communication between CPU and/or threads and management of task assignment, all this essential for a PGA. Actually, some researchers do not use such libraries and APIs to implement their algorithms, but they may help researchers to quickly put their algorithm into practice. Software and API aspects are described in Section 4.3. `Application` focuses in the final application and deals with features typical of this real world application, while `Problem Domain` classifies PGAs according to the fundamental aspects of the solved problem. `Problem Domain` is distinguished from the viewpoint of the number of objectives, its fitness dynamics, the number of constraints, solution representation, and where an instance is coming from. Applications and problem domain are discussed in Sections 4.4 and 4.5.

Regarding the implementation of a PGA, there exists mainly four kinds of PGA models: global parallelization, island model, cellular model, and hybrid models. The following subsections explain these details.

## 2.2 Global Parallelization

The global parallelization is the most simple implementation of a PGA. Fig. 2a illustrates the global model. In the figure, gray circles indicate solutions. In the global model, tentative solutions are managed a centralized

population, and each solution is evaluated in parallel. After evaluating solutions, new tentative solutions are generated from the centralized population. The global model is generally classified into two further approaches: generational and steady-state. The generational approach generates a new population after all slave nodes completed their tasks. On the other hand, the steady-state approach generates a new solution immediately after one slave node completes its task and returns it as a result. Unlike the generational approach, that replaces one complete generation by another, the steady-state approach does not wait all solution evaluations, and generates and replace a solution one-by-one. In steady-state mode of operation the GA merges new and old solutions continuously, yielding a usually faster convergence of the population with respect to generational models. Though this happens in global parallelization, it is also the main existing distinction for normal panmictic GAs where parallelism is not present.

The well-known master-slave parallel architecture is generally employed to implement the global model, where the master computing node manages the population and sends evaluation task to slave computing nodes. The slave computing nodes execute tasks given by the master node and return the expected results, e.g., the fitness of a received solution, or the resulting individual after altering it in some way (mutation done in parallel for example). Tasks handled with the slave nodes are usually the evaluation of solutions because it is the most computationally expensive part in the GA procedure. In addition to the evaluation, variation operators are often executed on the slave nodes. Since the global model does not change the numerical behavior of a sequential (panmictic) GA, the same search ability can be expected between a sequential and global parallel GAs (of course faster in the case of the parallel GA).

### 2.3 Parallel Island Model

The parallel island (or multi-population) model is a PGA that manages several sub-populations in separate islands, and executes the GA procedure in each island in parallel over different set of solutions. Fig. 2b shows a picture of the island model. Unlike the global parallelization manages a centralized population, the island model manages a set of decentralized populations. The decentralized populations are evolved separately and information of solutions in each island are exchanged at certain moments of evolution. This exchange of solutions is called a migration. The migration moves or exchanges (usually a copy of) solutions of each island to other ones depending on the topology of islands. Several topologies have been proposed, for example, ring, hypercube, mesh, random, and fully connect. The topology affects how migrated solutions are propagated to other islands and thus global convergence to the optimum. Regarding the migration, migration frequency and selection/replacement strategies must be configured. The migration frequency determines the interval of the migration, while the selection/replacement strategies determine which solutions of each island are selected as the migrants or replaced with the migrants from other islands. Needless to say, all islands can follow a similar strategy for search (e.g., population size, probability of genetic operators) and static configurations. On the other hand, to increase the diversity of solutions or to promote the convergence speed, heterogeneous policies and/or dynamic configurations can be employed. In the homogeneous structure, all islands have the same strategy for search, i.e., run the same algorithm in all islands, while in the heterogeneous structure, each island has different strategies, for example, different population size, and/or different probability of genetic operators [59], and in some heterogeneous structures, each island perform different EAs [34]. According to this, optimization behavior of each island is differ from each other, which contributes to increase the diversity of solutions. Dynamics of configuration is another aspect of the island model. In the static configuration, configuration of search is constant during the optimization process, while in the dynamic configuration, configuration may change during the optimization process depending on the pre-defined manner or depends on quality of solutions in each island.

To implement the parallel island model, each island is generally assigned to one computing element. For example as shown in Fig. 2b, four CPUs are used to execute a parallel island model having four islands, each one

managed by separate CPU where a GA on different population is running. At every migration migration step, a form of solution information is exchanged (migrated) to other CPU following some sort of communication strategy, e.g., shared memory or message passing, according to migration topology.

## 2.4 Parallel Cellular Model

The parallel cellular (or fine-grain) model arranges solutions on a grid structure and the variation procedure and the replacement are executed to solutions within a defined neighborhood. Fig. 2c illustrates the parallel cellular model. In the parallel cellular model, all solutions are arranged on a grid, and variation operators (e.g., crossover and mutation) and replacement are executed within a certain neighborhoods of a grid. For example in Fig. 2c, each solution is mated with solutions of top, bottom, left, and right neighborhoods on a grid, and a generated offspring is replaced within its birth neighborhood. A toroidal rectangle is mostly employed as a grid topology, while other topologies can be considered, such as 1-D or 3-D grid shape. Several neighborhood strategies can be employed, for example, Von Neumann shown in Fig. 2c is usually used, while a Moore neighborhood, in which solutions at top left, top, right, bottom left, and bottom right are neighbors in addition to the Von Neumann neighborhood, is also used. As it happens in the island model, non homogeneous structure can be employed in the cellular model, in particular, each solution on the grid can be managed according to different parameter settings and/or variation operators. Additionally, an updating strategy can be configured in the cellular model. There are synchronous and asynchronous update of cells [57]. Synchronous updating updates all solutions simultaneously, while asynchronous updates cells at a time in some order. And in asynchronous updating, several updating strategies for the positions to be updated exist, fixed line sweep, fixed random sweep, new random sweep, and uniform choice [105]. In fixed line sweep, cells are updated sequentially from left to right and line after line starting from upper left corner cell. In fixed random sweep, the next cell to be updated is chosen with uniform probability, and the same permutation is then used for all update cycle. While this, new random sweep generates new random permutation for each update. Uniform choice always randomly selected cells to be update with uniform probability.

Several implementations can be considered for the parallel cellular model. Fig. 2c shows an example of implementations, where four CPUs are employed and each CPU manages a grid including nine solutions. Four CPUs simultaneously execute selection, variation operators, and replacement following a defined neighborhood strategy. In this implementation, information of solutions on the edge of each grid have to be exchanged to each other through a physical communication network, because a part of neighborhoods is in a grid of other CPU. Other different and interesting implementation can be done when using a graphics processing unit (GPU); there, each solution is assigned to one core unit or thread in the GPU, and each processing unit executes selection, variation operators, and replacement for an assigned solution and its neighborhoods. Many issues happen here because of the synchronicity of GPUs and their need of making exactly the same operations if one wants to have a maximum speedup, so the say in which a cellular GA runs on a GPU is tricky [126].

## 2.5 Parallel Hybrid Model

The parallel hybrid approach combines two or more of the PGAs described before. We can see examples of hybrid models in Fig. 3. Fig. 3a shows a hybrid model of the island model and the global parallelization, in which the island model is used at a first level and each island executes the GA tasks of solution evaluations and/or variation operators in parallel. Fig. 3b shows a hybrid model of the two island models, in which the island model is also used at a first level and each island also executes the island model PGA. Fig. 3c shows a hybrid model of the island model and the cellular model, in which the island model is also employed as a base model, while each island executes the cellular parallelization. Note that these are just examples and other hybrid approaches of combinations of PGA are possible.

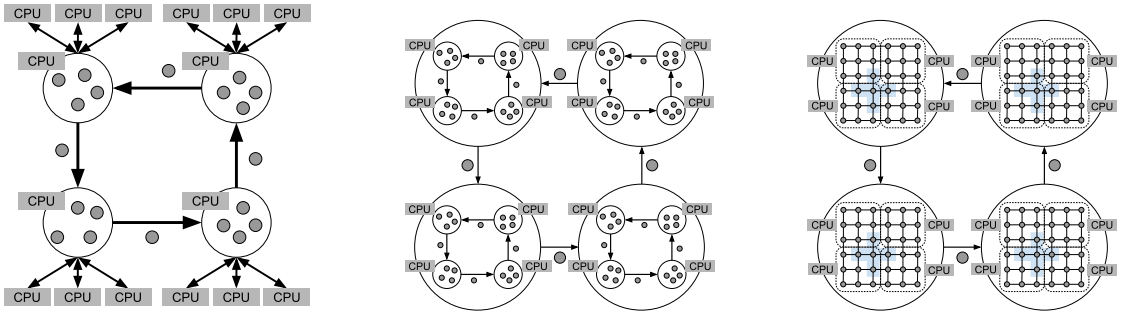


Fig. 3. Three examples of hybrid PGAs: (a) Island × Global, (b) Island × Island, (c) Island × Cellular

## 3 ANALYSIS OF RECENT PGA STUDIES

In this section, we show an analysis and offer statistics on a set of recent PGA studies. This survey considers papers related to PGAs from IEEE Xplore[1], ACM Digital Library[2], Springer Link[3], and Scopus[4]. These scientific web services contain most important journal papers and conference proceedings, not only in the EC domain, but also in the computer science and engineering domains. They were found after a careful search by using several keywords related to PGAs. In particular, we use the combination of keywords related to the algorithm itself (parallel, island model, island based, fine-grain, course-grain, master-slave, subpop, sub-pop), the hardware (cluster, GPU, graphics processing unit, FPGA, programmable gate array, multi-core, multicore) and the term "genetic algorithm". The search year was specified from January 1$^{th}$ 2013 to May 24$^{th}$ 2018. We totally correct 1,229 papers including 612 journal papers and 617 conference papers.

Section 3.1 firstly analyzes the statistics of the number of publications for each year, and identify the journals and conferences publishing more PGA articles in the timeframe analyzed. Then, Section 3.2 analyzes the keywords used by these PGA papers (a semantic study of the state of the art), while Section 3.3 discusses the authorship of the corrected papers (a strategic analysis of the state of the art). Finally, Section 3.4 summarizes all the findings of this first part of our survey, focusing in how PGA articles (and venues) developed themselves in the last years.

### 3.1 Publications

In this section, we discuss statistics of publications of PGA works. Firstly, we show the change of the number of publications in the recent five years. Then, we show the detail of publications from the viewpoints of journal publications and conference papers, and discuss what journals/conferences PGA researches are focusing on. This analysis helps new master and PhD researchers to find which journals/conferences should be considered and focused on to learn recent trend of PGAs and are suited to submit their new PGA papers.

Fig. 4 shows the number of publications related to PGAs appeared in both journals and conferences in the last five years. The horizontal axis indicates year, while the vertical axis indicates the number of publications for each year. The blue bar indicates the number of journal papers, while the orange bar indicates the number of conference papers and chapter in book. From this histogram, it can be found that almost 200 papers related to PGAs are published every year, for a total of 1,229 papers in the last five years. Note that publications in 2018 are

[1]https://ieeexplore.ieee.org/Xplore/home.jsp
[2]https://dl.acm.org/
[3]https://link.springer.com/
[4]https://www.scopus.com/wsearch/form.uri?display=basic

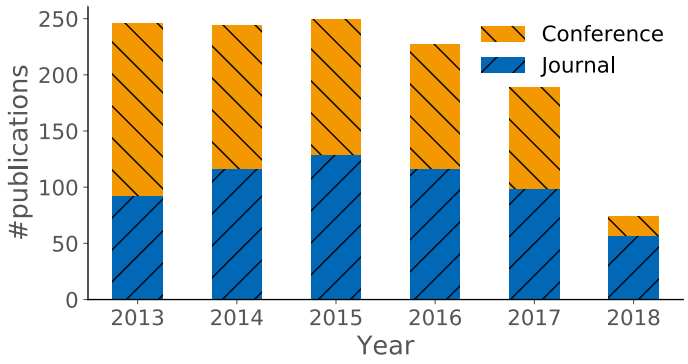


Fig. 4. The number of publications related to PGAs from 2013 to 2018

less than 100 because this survey has been done bit before summer of 2018. We can notice that the total number of papers has slightly decreased in the last two years, what we interpret as a shift to interest to the applications of PGAs instead of going to the fundamentals of these algorithms. This is because many researchers recently pay attention to applications of PGA, but not to fundamental methodology of it, and their titles, keywords, and abstracts do not contain words related to PGA explicitly.

Fig. 5 shows the journal papers that have the top 30 number of publications, and the horizontal axis indicates the name of journals, while the vertical axis indicates the number of publications. We can see mainly three groups of journals from this figure. The first group consists of the top three number of publications. Two of them, Concurrency Computation: Practice and Experience, and International Journal of Distributed Sensor Networks, are related to parallel systems while one of them, Applied Soft Computing Journal, is related to soft computing including GA. This indicates many of PGA journal papers were submitted as topics of parallel systems and soft computing algorithms. The second group, from $4^{th}$ to $11^{th}$ ranks, contains journals related to applications or manufacturing. Actually, six journals out of these eight ones are related to applications or manufacturing, in particular, International Journal of Applied Engineering Research ($4^{th}$ rank, 8 publications), Neural Computing and Applications ($5^{th}$ rank, 7 publications), Computers and Industrial Engineering ($6^{th}$ rank, 6 journals), ARPN Journal of Engineering and Applied Sciences ($6^{th}$ rank, 6 publications), Expert Systems with Applications ($6^{th}$ rank, 6 publications), and Journal of Intelligent Manufacturing ($6^{th}$ rank, 6 publications). This indicates that many applications of PGAs are published in journals related to applications and manufacturing, and they are submitted to journals that well match to topic of specific applications. The third group, from $12^{th}$ to $30^{th}$ ranks, has the similar trend, many journals are related to applications, such as Automation of Electric Power Systems ($12^{th}$ rank, 4 publications), IEEE Transactions on Industrial Informatics ($12^{th}$ rank, 4 publications), IET Signal Processing ($12^{th}$ rank, 4 publications), and Multimedia Tools and Applications ($12^{th}$ rank, 4 publications). Since journals regarding applications, industrial, and manufacturing are ranked in the top 10 list, it seems we can suggest that there is a recent shift in attention to applications of PGAs rather than their fundamentals. In particular, IEEE Transactions on Evolutionary Computation, IEEE Transaction on Parallel and Distributed System, and the Journal of Parallel and Distributed Computing, well-known journals in the EC and the parallelism domains, have few number of publications related to PGAs, in detail, 1, 2, 4 papers, respectively.

Table 1 shows the details for the top 10 journals ($10^{th}$ rank is a tie). "Journal" column indicates the name of journal, while "Total" one indicates the total number of publications for each journal; and the rest of columns

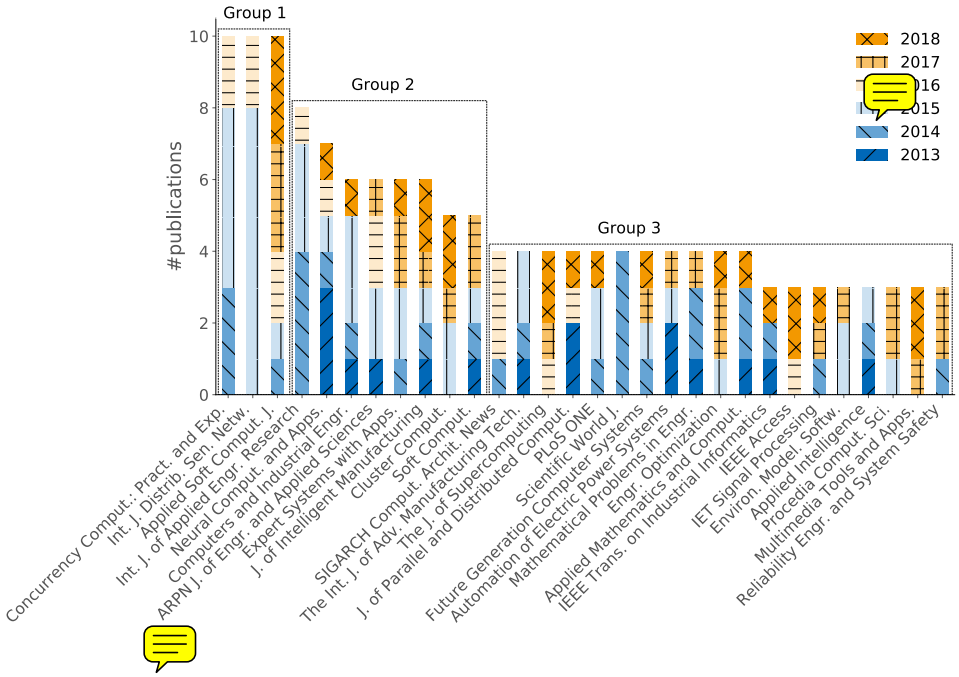


Fig. 5. The number of journal papers from 2013 to 2018 (the top 30 number of publications)

Table 1. Top 10 journals ($10^{th}$ rank is tie)

| Journal | Total | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|---|---|---|
| Concurrency Computation: Practice and Experience | 10 | 0 | 3 | 5 | 2 | 0 | 0 |
| International Journal of Distributed Sensor Networks | 10 | 0 | 0 | 8 | 2 | 0 | 0 |
| Applied Soft Computing Journal | 10 | 0 | 1 | 1 | 2 | 3 | 3 |
| International Journal of Applied Engineering Research | 8 | 0 | 4 | 3 | 1 | 0 | 0 |
| Neural Computing and Applications | 7 | 3 | 1 | 1 | 1 | 0 | 1 |
| Computers and Industrial Engineering | 6 | 1 | 1 | 3 | 0 | 0 | 1 |
| ARPN Journal of Engineering and Applied Sciences | 6 | 1 | 0 | 2 | 2 | 1 | 0 |
| Expert Systems with Applications | 6 | 0 | 1 | 2 | 0 | 2 | 1 |
| Journal of Intelligent Manufacturing | 6 | 1 | 1 | 1 | 0 | 1 | 2 |
| Cluster Computing | 5 | 0 | 0 | 2 | 0 | 1 | 2 |
| Soft Computing | 5 | 1 | 1 | 1 | 0 | 2 | 0 |

indicates a breakdown for each year. From this table, it is firstly indicated that three journals, Concurrency Computation: Practice and Experience, International Journal of Distributed Sensor Networks, and International Journal of Applied Engineering Research, publish PGA papers all in one go from 2014 to 2016, while no paper is published in 2013, 2017, and 2018. On the other hand, other journals constantly publish one or more PGA papers every year. This indicates that although the number of journal publications of PGA works is not many, PGAs are

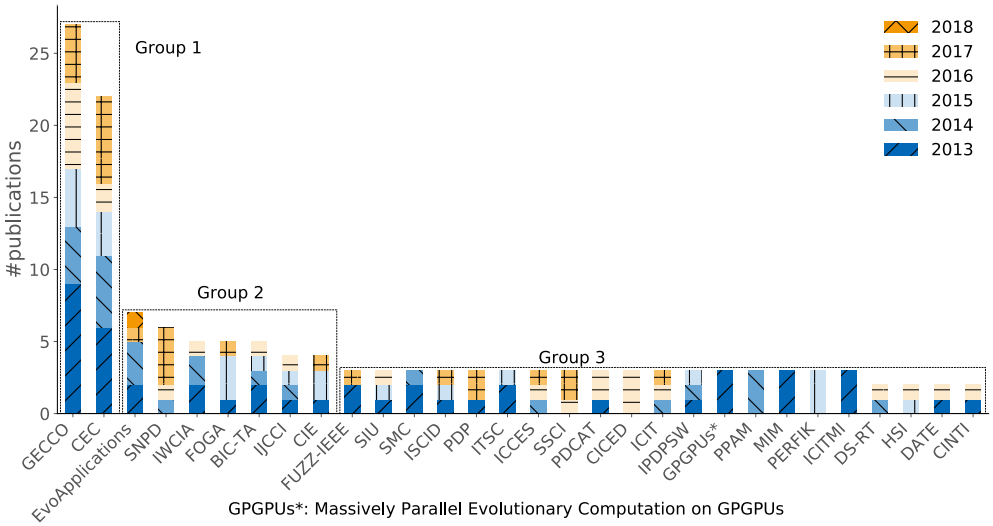


Fig. 6. The number of conference papers and chapters in books from 2013 to 2018 (top 30)

Table 2. Top 10 conferences ($10^{th}$ rank is tie)

| Conference/Chapters in book | Total | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|---|---|---|
| GECCO | 27 | 9 | 4 | 4 | 6 | 4 | 0 |
| CEC | 22 | 6 | 5 | 3 | 2 | 6 | 0 |
| EvoApplications | 7 | 2 | 3 | 0 | 0 | 1 | 1 |
| SNPD | 7 | 0 | 1 | 0 | 2 | 4 | 0 |
| BIC-TA | 5 | 2 | 1 | 1 | 1 | 0 | 0 |
| FOGA | 5 | 1 | 0 | 3 | 0 | 1 | 0 |
| IWCIA | 5 | 2 | 2 | 0 | 1 | 0 | 0 |
| CIE | 4 | 1 | 0 | 2 | 0 | 1 | 0 |
| IJCCI | 4 | 1 | 1 | 1 | 1 | 0 | 0 |
| 18 conferences are the | 3 | - | - | - | - | - | - |

18 conferences with similar #papers* CICED, FUZZ-IEEE, ICCES, ICIT, ICITMI, IPDPSW, ISCID, ITSC, Massively Parallel Evolutionary Computation on GPGPUs, MIM, PDCAT, PDP, PERFIK, PPAM, SIU, SMC, SocProS, SSCI

constantly payed attention in the recent five years not only in the soft computing or the parallel system domain, but also in the domain of applications.

Fig. 6 shows the papers in the conference proceedings and chapter in book that have the top 30 number of publications. Axes and colors have the same meaning as Fig. 5. From this figure, we can see also three groups of conferences. The first group consists of two conferences, ACM Genetic and Evolutionary Computation Conference (ACM GECCO) and IEEE Congress on Evolutionary Computation (IEEE CEC), which definitely have large number of publications in the PGA domain. Actually, these two conferences have high impact not only in the PGA domain,

but also in the whole field of evolutionary computation, so that it is still important for PGA researcher to gather information from and present new result to these two major conferences. The second group consists of seven conferences, EvoApplications, SNPD, BIC-TA, FOGA, IWCIA, CIE, and IJCCI. Notable here is that five conferences out of seven, EvoApplications, SNPD, BIC-TA, IWCIA, and CIE, contains the term of "applications" in the name of conferences and focus on an area of applications of evolutionary computation and computational intelligence including PGA. This indicates that the PGA works applied to real world problems have been published in such conferences that focus on applications of evolutionary computation and computational intelligence. The resting of this group, FOGA and IJCCI, are both related to GA and computational intelligence. The indication from here is that no conference directly related to the parallel system exists in these two groups up to 9$^{th}$ rank and most of PGA works have been published conferences related to evolutionary computation and computational intelligence. In contrast to the previous two groups, the third group consists of conferences that relate to the parallel system, although it, of course, consists of ones related to the soft computing and computational intelligence. In particular, five conferences, IPDPSW, Massively Parallel Evolutionary Computation on GPGPU (which is chapter in book), PDCAT, PDP, and PPAM mainly focus on the parallel system, but not the evolutionary computation. This indicates that some PGA researches are payed attention to from the point of view of the parallel system.

Table 2 shows the details of top 10 conferences and chapter in book, each column has the same meaning as Table 1. From this analysis, it can be confirmed that two major conferences in the EC domain, ACM GECCO and IEEE CEC, have high importance for the PGA domain (as one could expect). Actually, these two conferences, ACM GECCO and IEEE CEC, account for 4.5% and 3.7% of publications in all conference papers related to PGA, respectively. In both conferences, PGA topics are constantly presented every year.

## 3.2 Keywords

This section discusses on keyword used in PGA papers. We create a word cloud of keywords in PGA papers and discuss on hot topics of recent PGA works by analyzing what keywords are frequently used.

Fig. 7 shows the word cloud we have generated by using keywords in the corrected papers. The plural and case forms of the terms in the cloud are filtered, so that "genetic algorithm", "Genetic Algorithm", and "genetic algorithms" are all treated as "genetic algorithm". Larger words in the word cloud are the ones used more frequently in the papers analyzed.

From this word cloud, although keywords related to algorithm such as "genetic algorithm", "optimization" or "parallel genetic algorithm" are understandably used more frequently, it is notable that also "scheduling" is frequently used as a keyword in PGA papers. This is because many applications of scheduling can be found as applications (for example, transportation, manufacturing or task assignment) and even internal ways of implementing PGAs. The keyword "multi-objective" is attracted researchers attention as problem domain of PGA. In the last decade, since researches in multi-objective GA (MOGA) increased in the EC domain, their parallelization also became an important topic. As a keyword related to hardware, the terms related to GPU, such as "graphics processing unit" and "cuda" are also frequently found. GPU recently becomes a very important device, not only in the EC research, but also for other domains like deep learning. The reason is the big computation power that can be drawn from them to compute many tasks simultaneously.

## 3.3 Authorship

In this section, we analyze the authorship of PGA researches. Firstly, we discuss the change of the number of authors in PGA researches in the last five years. Additionally, we show the percentage of new comers in every year. Then, the number of works per each author and the number of co-authors in each PGA paper are shown. And finally, we show countries where authors of PGA research belong to and their relationship from the point of view of co-authorship.

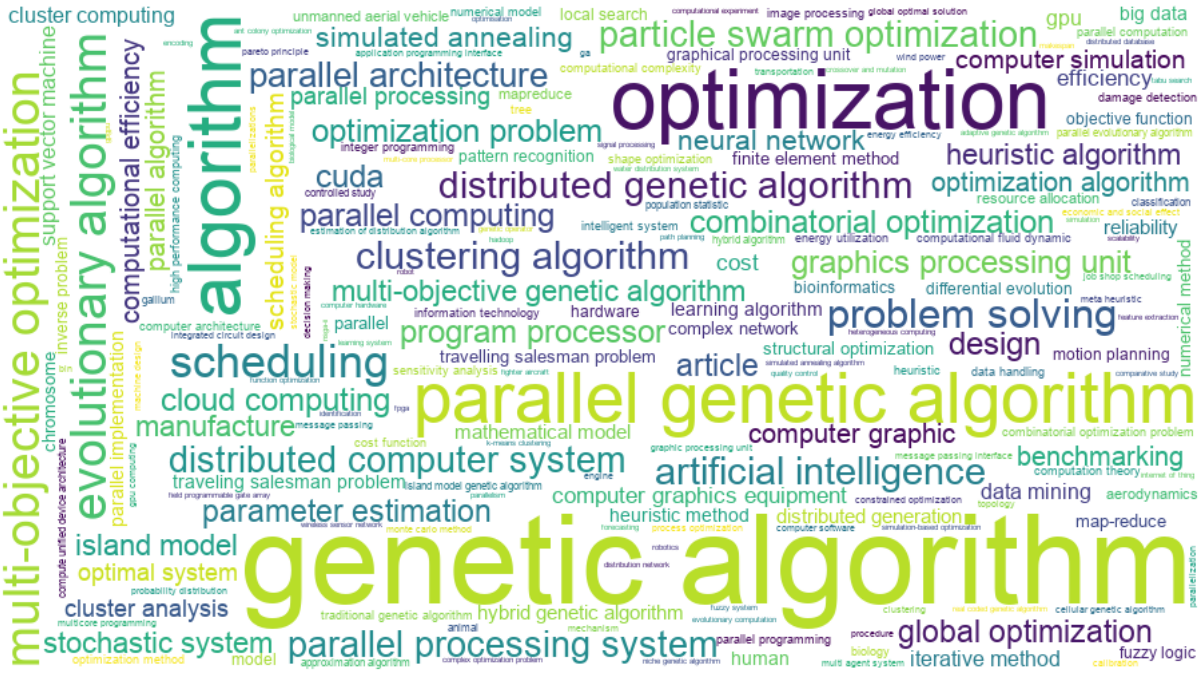


Fig. 7. Word cloud of keywords in PGA papers

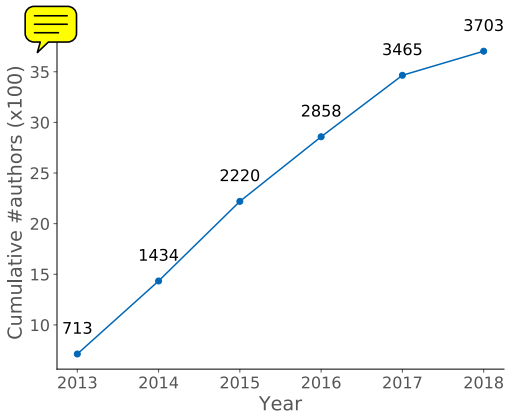


Fig. 8. Cumulative number of authors from 2013 to 2018

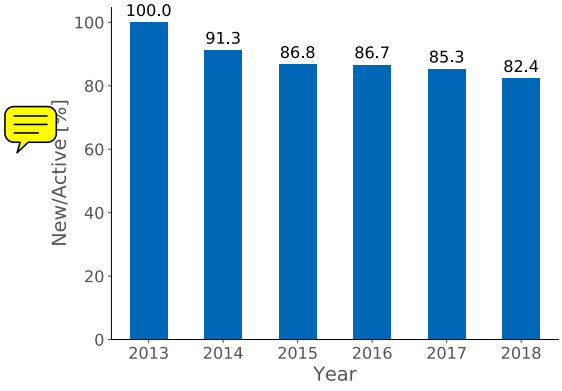


Fig. 9. Percentage of new authors each year from 2013 to 2018

Fig. 8 shows the cumulative number of authors who publish PGA papers from 2013 to 2018. The horizontal axis represents year, while the vertical axis shows the cumulative number of authors. From this figure, the cumulative number of authors constantly increases every year from 2013 to 2017, and in total, 2,825 authors one or more

Table 3. The number of authors who published each number of works and its percentage

| # Works | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| # Authors | 2453 | 293 | 47 | 18 | 7 | 3 | 1 | 3 |
| % | 86.83% | 10.37% | 1.66% | 0.64% | 0.25% | 0.11% | 0.04% | 0.11% |

Table 4. The number of co-authors for each paper and its percentage

| # Co-authors | 1 | 2 | 3 | 4 | 5+ |
|---|---|---|---|---|---|
| # Works | 70 | 287 | 282 | 206 | 181 |
| % | 6.82% | 27.97% | 27.49% | 20.08% | 17.64% |

PGA papers from 2013 to 2017. Although the increase of the cumulative number of authors from 2017 to 2018 is slow, this is because this survey is conducted in the middle of 2018.

Fig. 9 shows the percentage of new authors to active one for each year from 2013 to 2018. The horizontal axis represents year, while the vertical axis shows the percentage of new authors to active ones for each year. Here, "new author" counts authors who firstly publish PGA paper(s) in the year during 2013 to 2018, while the number of "active author" counts authors who publish PGA paper(s) either they are new or not. Since all authors are new and active ones in 2013 in this survey, the percentage in 2013 is 100%. In 2014-2018, the percentages of new authors to active ones are relatively high, in particular 83-91%. This indicates that most of authors newly join in the PGA researches and they publish their work only once or a few times.

Table 3 shows the number of authors that published each number of works. "# Authors" row indicates the number of authors who publish papers of the number of corresponding column, while "%" row indicates its percentage. 86.83% authors published only one PGA paper, while 10.37% ones published two papers from 2013 to 2018, in total 97.20% of authors published less than or equal to two PGA papers. This analysis supports the result of Fig. 9 in which most of authors are newcomer to the PGA domain.

Table 4 shows the statistics of authorship for each paper. "# Co-authors" row indicates the number of authors per paper, while "# Works" and "%" rows indicate the number of works with authors of the number of corresponding column and its percentage. From this table, it is shown that most of works are done by two or three authors, while 17.64% of works are done by five or more authors.

Table 5 shows the number of authors from each country and region. "Country" column indicates countries, while "# authors" and "%" columns indicate the number of authors from a country corresponding to each row and its percentage. China is a country where the most number of authors are in the PGA domain, and its percentage is about 34% of all authors. Subsequently to China, authors from India and USA account for about 7.5% of all authors. Subsequently to them, authors from Japan, Spain, Brazil, Iran, Taiwan, UK, Canada account for 2–4% of all authors, and totally about 68% of authors are from these countries and regions.

Fig. 10 shows the relationship between countries from the viewpoint of co-authorship of the PGA papers. The size of circles indicates the number of authors, while the width of edges indicates the number of co-authorships between two countries. The color of circles indicates the difference of communities of co-authorship network. From this analysis, the most frequent co-authorship of two countries are China and USA, and these countries have many co-authorship with several countries regardless of geographical zone. Other than this, several communities of co-authorship of countries can be found. For example, Spain has connections to many countries in Europa and South America, such as Brazil, Portugal, Argentina, Colombia, Luxembourg, Finland, and so on. Focusing on India, there are many co-authorships between Asian countries, for example South Korea, Taiwan, Singapore, Malaysia, and so on. In Europa, UK, France, and Germany work with authors of many countries not only in Europa but also in Asia, North America, and South America. From these analysis, in the PGA domain, there are

Table 5. The number of authors from each country and region (top 10)

| Country | # authors | % |
|---|---|---|
| China | 1020 | 33.81% |
| India | 230 | 7.62% |
| USA | 224 | 7.42% |
| Japan | 123 | 4.08% |
| Spain | 99 | 3.28% |
| Brazil | 84 | 2.78% |
| Iran | 82 | 2.72% |
| Taiwan | 72 | 2.39% |
| UK | 72 | 2.39% |
| Canada | 62 | 2.06% |

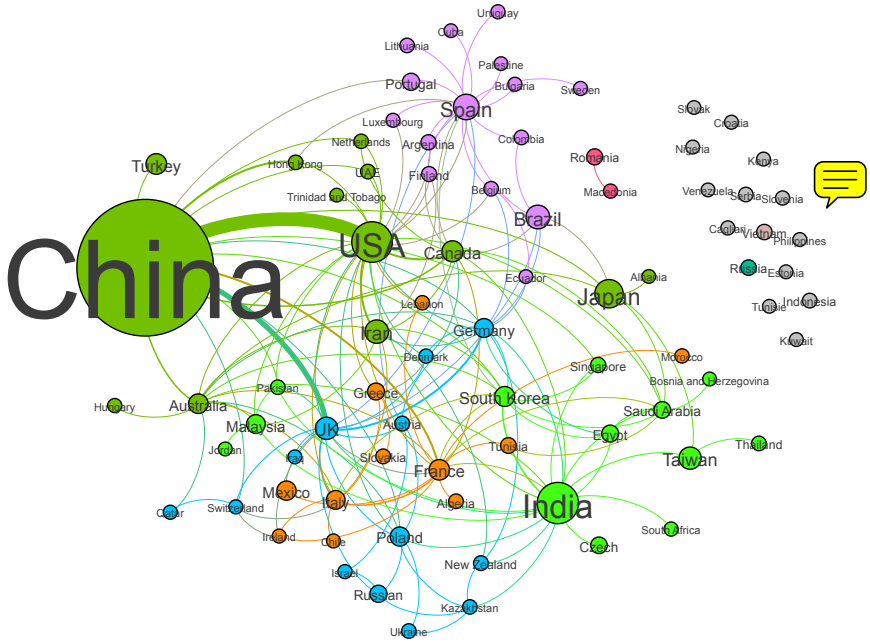


Fig. 10. Country relationship from the viewpoint of co-authorship

many co-authorship of researchers not only limited in a certain countries, but also different geographical zones and many relations of countries.

### 3.4 Summary

In this section, we showed an analysis and offered statics on a set of recent PGA studies. We analyzed publications of PGAs from three points of view, the number of publications through year, keyword used in papers, and authorship. This section is summarized as follows:

- The number of publications related to PGAs decreases in the last 5 years. Additionally, many researchers pay attention to applications of PGAs, but not basics or fundamentals.
- From the analysis of keywords appeared in PGA papers, it is indicated that "scheduling", "multi-objective", and "graphics processing unit" are hot topics in this domain.
- More than 3,000 researchers joined in the PGA domain and more than 80% of researchers are new ones in the last 5 years. Most works are collaborate one with two or more authors. And many works are treated not only by authors in one country but also by collaboration of several countries.

## 4 LITERATURE REVIEW

In this section, we show the literature review. In particular, we focus on papers that are cited more than 20 papers. Table 6 shows a list of papers that are cited more than 20 papers[5]. In this table, "Ref." column indicates the reference of papers, while "Cites" column indicates the citations of each paper. "Impl." column indicates the employed implementation of PGA in each paper. "HW" column indicates hardwares used to implement PGAs, while "# processors" shows the maximum number of processors actually used in the experiment. "API", "Library" and

[5]The number of citations is checked in Scopus at 24 May 2018.

"Prog. lang." columns shows APIs, Libraries, and programming languages used to implement PGAs, respectively. "Prob. instance" finally indicates problem instances employed in papers, in which "Academic" represents academic standard benchmark problem instances, "Problem generation" represents problem instances randomly generated, and "Application" represents that PGA is applied to real-world applications in that paper. If such information is not clearly indicated or not available in the paper, a cell is filled with "-".

Section 4.1 discusses the implementation of PGAs employed and studied in the listed papers. Sections 4.2 and 4.3 describe hardware and software employed in the listed papers, respectively. Section 4.4 describes academic problem instances and problem generators, while Section 4.5 describes real-world applications solved by PGAs. Finally, Section 4.6 summarizes this section.

## 4.1 Parallel implementation

Firstly in this section, we discuss on papers from the point of view of the PGA implementation we described in Section 2. From the implementation aspect, the parallel island model and the global parallelization are majority, 16 papers out of 20, while one papers employs parallel cellular model. Four papers employ parallel hybrid approach. These details are described in the following.

**Parallel island model:** Table 7 summarizes the characteristics of the parallel island models used in the papers in Table 6. "Topology" column indicates topology used in that paper, while "Migration Frequency" column indicates how frequently solution migration is conducted. "Selection" and "Replacement" columns indicate how to select migrated solution(s) from subpopulation and) and how to select replaced solution with migrated solution(s) from other subpopulation, respectively. "Configuration" column indicates whether configuration of genetic parameters, e.g., mutation rate and migration frequency, are static or dynamically changed during evolution process. "Structure" column indicates each subpopulation has uniform (same) structure or nonuniform one, e.g., different mutation rates or different migration strategy.

Most papers employ all connected topology where all subpopulations are connected each other and immigrants move to other all subpopulations. Osaba et al. [86] employs characteristic *one-by-one* topology. They propose a novel meta-heuristic, named Golden Ball (GB), to solve combinational optimization problems [86]. GB is based on soccer concept and similar to the island model PGA, in which there are several teams (corresponding to islands) and they compete each other. Their team players (corresponding to solutions) are trained between matches and some team members are translated to other team (corresponding to migration) every end of season. The essential differences between general PGAs and GB are; (1) all teams (islands) are ranked depends on the quality (fitness) of assigned players; and (2) teams with higher rank can get good player (solution) from ones with lower rank, in particular, $i^{th}$ ranked team can get $i^{th}$ best player from $i^{th}$ worst team. This means there exists hierarchy of islands depends on assigned solutions and migration is performed nonuniformly.

Several migration frequencies are used, some papers migrate solutions every generations, while others uses the migration frequency based on the number of generations. The work of Porta et al. [90] employs wall-clock time as the migration frequency, independent to the number of generations. The work of Kurdi [73] decides the migration if there are no improvement in all of the islands.

For the selection strategy, four papers employs the best solution to be migrated, while for the replacement strategy, three papers employs the worst solution to be replaced with migrated solutions. In the work of Roberge et al. [99], populations of all islands are merged and new populations are generated by randomly permutating and splitting the merged population. The work of Porta et al. [90] constructs new population with best solutions from each island and all islands start from this new population. In the work of Kurdi [73], a newly naturally inspired migration strategy is proposed in which *MigCou* worst individual (*MigCou* is the count of migration occurs) in each island are selected to be migrated to other islands, unlike the classical island models generally select best individuals.

Table 6. Papers cited more than 20 times

| Ref. | Cites | Impl. | HW | # processors | API | Library | Prog. lang. | Prob. instance |
|---|---|---|---|---|---|---|---|---|
| Roberge et al. [99] | 192 | Island | Multicore CPU | 8 | Parallel Computing Toolbox | - | MATLAB | Real-world |
| Liu et al. [76] | 49 | Global | Cluster | 30 | MPI | - | Fortran | Real-world |
| Devos et al. [45] | 45 | Global | Cluster | 16 | Distributed Computing Server | GEATbx | - | Real-world |
| Ishibuchi et al. [67] | 34 | Island | Multicore CPU | 8 | - | - | - | Academic |
| Porta et al. [90] | 33 | Global<br>Island<br>Hybrid | Mulcicore CPU<br>Cluster<br>Hybrid | 8<br>8<br>32 | -<br>java.util.concurrent<br>MPJ<br>MPJ | - | Java | Real-world |
| Sahingoz [101] | 31 | Global | Multicore CPU | 8 (with HT) | - | - | Java | Real-world |
| Shayeghi et al. [107] | 30 | Hybrid | - | (up to) 240 | - | | - | Real-world |
| Roberge et al. [100] | 28 | Hybrid | GPU | 512 | CUDA | - | - | Real-world |
| Osaba et al. [86] | 27 | Island | - | - | - | - | - | Academic |
| Hong et al. [65] | 25 | Global | Cluster | (up to)130 | - | - | Java | Problem generation |
| Yang et al. [131] | 24 | Global | Cluster | 48 | Node.js, HTCondor | SiPESC.OPT | JavaScript | Real-world |
| Tosun et al. [120] | 24 | Island | Cluster | 240 | OpenMPI | - | C++ | Academic |
| Allahviranloo et al. [18] | 23 | Island | Cluster | 12 | Parallel Computing Toolbox | - | MATLAB | Problem generation |
| Pinel et al. [88] | 23 | Cellular | GPU | 448 | CUDA | - | - | Problem generation |
| Dorronsoro et al. [48] | 22 | Island | Multicore CPU | 8 | - | jMetal | Java | Problem generation |
| Kurdi [73] | 21 | Island | Multicore CPU | 3 | - | - | - | Academic |
| Cekmez et al. [31] | 20 | Global | GPU | 384 | CUDA | - | C | Academic |
| Cao and Ye [30] | 20 | Island | Multicore CPU | 4 | - | - | - | Real-world |
| Park et al. [87] | 20 | Global | Cluster | 64 | Distributed Computing Server | - | MATLAB | Real-world |
| Davis et al [39] | 20 | Hybrid | Cluster | - | - | | Python | Real-world |

"-" in the cell in this table indicates that this information is not available in the corresponding paper

Table 7. Type of island model used in the papers cited more than 20 times

| Ref. | Topology | Migration Frequency | Selection | Replacement | Configuration | Structure |
|---|---|---|---|---|---|---|
| Roberge et al. [99] | Full connect | 10 times | Random permutation of a global population | | Static | Uniform |
| Ishibuchi et al. [67] | Ring | {20, 50, 100, 200} generations | Best | Worst | Static | Nonuniform |
| Porta et al. [90][a] | Full connect | {3, 6, 15} minutes | New population with best solutions from each island | | Static | Nonuniform |
| Osaba et al. [86] | One-by-one | The number of subpopulations | $i^{th}$ best[b] | $i^{th}$ worst | Static | Nonuniform |
| Tosun et al. [120] | Full connect | Every generation | Best | No replacement | Static | Uniform |
| Allahviranloo et al. [18] | Full connect | Every generation | Best | Worst | Static | Uniform |
| Dorronsoro et al. [48] | Full connect | Every generation | Best | No replacement | Static | Nonuniform |
| Kurdi [73] | Full connect | Adaptive[c] | *MigCou*$^{th}$ worst[d] | Worst | Dynamic | Nonuniform |
| Cao and Ye [30] | Full connect | 10 generations | - | - | Static | Uniform |

[a] Cluster version
[b] $i$ is rank of a subpopulation
[c] Migrate if there are no improvements in all of the islands for ten successive generations
[d] *MigCou* is the count of migrations occurred

Table 8. Type of global parallelization used in the papers cited more than 20 times

| Ref. | Reproduction | Slave's task |
|---|---|---|
| Liu et al. [76] | Generational | Evaluation |
| Porta et al. [90][a] | Generational | Selection, crossover, mutation, evaluation |
| Sahingoz [101] | Generational | Selection, crossover, mutation, evaluation |
| Hong et al. [65] | Generational | Evaluation |
| Yang et al. [131] | Generational | Evaluation |
| Cekmez et al. [31] | Generational | Selection, crossover, mutation, evaluation |
| Park et al. [87] | Generational | Evaluation |

[a] Multicore CPU version

Some papers employs nonuniform configuration of islands, i.e., different configuration is applied to each island. The work of Ishibuchi et al. [67] applies the island model PGA to genetic based machine learning (GBML) to optimize fuzzy rules. In their model, each island does not only manage different subpopulation, but also has different data set to be learned (nonuniform data set). After learning several generations, solutions migrates to other island, while a part of data set is also migrated in the reversed manner to the solutions migration. In the work of Kurdi [73], each island employs a different self-adaptation method as nonuniform configuration. The work of Dorronsoro et al. [48] proposes island based cooperative coevolution multi-objective evolutionary algorithms (CCMOEAs). As multi-objective variants, they employs Non-dominated Sorting Genetic Algorithm II (NSGA-II) [42], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [133], and Multi-objective Cellular Genetic Algorithm (MOCell) [68] In the proposed model, each subpopulation optimizes partial solutions, and best partial solutions are migrated to other island and are utilized to evaluate partial solutions.

**Global parallelization:** Table 8 summarizes the feature of the global parallelization used in the papers in Table 6. "Reproduction" column indicates whether the generational (synchronous) reproduction or the steady-state (asynchronous) reproduction is used. "Slave's task" column indicates what slave nodes of the master-slave implementation executes in the evolution process.

All papers use the generational (synchronous) approach. This is because the generational approach can be directly designed based on conventional non-parallel GAs and synchronous communication between master and slave nodes is easily implemented. As a characteristic of the generational approach, many cores or processors are used rather than the island model from Table 6. Since the generational approach has high scalability when the number of executions of solution evaluations and/or genetic operations increases by increasing the number of processors, it is suitable for parallelization on the system with many processors.

In the work of Liu et al. [76], the global parallelization is used to evaluate solutions by performing a generalized logit-based stochastic user equilibrium traffic assignment, a framework for the route choice problem. A HPC system in the Civil and Environmental Engineering department at the National University of Singapore is used in this paper and the maximum 30 processors are used to parallelize the solution evaluations.

The work of Sahingzo [101] also solves the UAV path finding problem on PGA. In this work, paths of multiple UAVs are optimized simultaneously. The global parallelization is employed and each processor of parallel computing node performs selection, crossover, mutation and local search. This work employs multicore CPU with four cores and four different UAV paths are optimized.

Hong et al. [65] also applies PGA to data mining task with fuzzy-based GA. They employ the master-slave implementation where fitness evaluation of each fuzzy-rule with 10,000 transactions, which is the most time-consuming process.

The work of Yang et al. [131] proposes a web-based parallel GA optimization framework based on high throughput distributed computation environment. The global parallelization of NSGA-II is implemented on distributed computation environment by using the Condor computation environment. On this framework, the

Table 9. Hybrid approaches used in the papers cited more than 20 times

| Ref. | Hybrid | Detail |
|---|---|---|
| Porta et al. [90][a] | Island×Global | Islands are fully connected, and each island is evolved with generational global parallelization |
| Shayeghi et al. [107] | Global×Evaluation parallel | Pool based steady-state global parallelization is employed, and a newly generated solution is evaluated in parallel |
| Roberge et al. [100] | Island×Evaluation parallel | Islands are connected with bidirectional ring topology, and each solution in island is evaluated in parallel |
| Davis et al. [39] | Global×Evaluation parallel | Pool based steady-state global parallelization is employed, and a newly generated solution is evaluated in parallel |

[a] Hybrid version

computation of the individual solution fitness value is enveloped as a HTCondor job. The optimization server is implemented with Node.js and JavaScript. Users can consume optimization services through internet by different browsers and facilities, such as, Chrome, Internet Explorer, Firefox, and so on. Moreover, particular client pages can be constructed accounting to users' requirements.

The work of Cekmez et al. [31] presents efficient implementation of GA on GPU to solve TSP, in particular a 1-thread in 1-position approach is developed to improve the performance through maximizing efficiency. Concretely, each thread of GPU generates new solution through tournament selection, crossover, mutation, and local search. The proposed method is implemented using CUDA on GPU with 384 cores. The proposed method is tested on the TSP instances from TSPLIB, and is compared with the serial version on CPU implementations. The experimental results show that the GPU version has a speedup from 366 to 1955× in comparison with the serial version.

The work of Park et al. [87] proposes distributed NSGA-II, a multi-objective genetic algorithm, to seismic retrofit design of 2D steel frame structure and 3D irregular reinforced concrete structure. Since a time-consuming nonlinear structural analysis is needed for the evaluation of seismic performances of a retrofitting design, they employ the global parallelization to evaluate solution in parallel. They use the cluster architecture, in which 16 PCs with quad-core processor are connected through network switching. In total this cluster has 64 processors. Distributed Computing Server provided by MathWorks is used. It is revealed that the distributed algorithm on a cluster of multicore PCs can reduce the computational time for the optimization without deteriorating the quality of the optimal solutions. In particular, although the efficiency decreases by increasing the number of processors, they achieve 75% efficiency with 64-core configuration.

**Parallel cellular model:** In this survey, only one parallel cellular model PGA with more than 20 citations is found. The work of Pinel et al. [88] proposes a new parallel design of cellular GA for GPU architecture, named as GraphCell. The population is initialized with random solutions except for one, which is the result of the Min-min heuristic, a simple deterministic algorithm proposed by Ibarra and Kim in 1977 for the scheduling problem of independent tasks. The recombination operator is run with one thread per task of a solution, unlike the usual design to evolve one solution by a single GPU thread. This means each solution of the population is recombined in parallel. They use GPU holding 14 multi-processors of 32 cores each, in total 448 processors, and employs CUDA to implement the proposed method. The proposed method is tested on problem instances of the independent task assignment problem ranging from 512 tasks over 16 machines, to 65,536 tasks over 2048 machines. The problem instances are randomly generated with high task and machine heterogenity. The population is set to $8 \times 8$, and the neighborhood used is von Neumann. The performance of the proposed method is clearly higher with respect to the CPU version when the problem size increases, especially a speedup ratio is 538 for the biggest instance.

**Parallel hybrid:** Table 9 summarizes the characteristics of the parallel hybrid models used in the papers in Table 6. "Hybrid" column indicates what kinds of implementation is combined, while "Detail" column describes an explanation of an implementation.

The work of Porta et al. [90] considers three architectures of parallelism: multicore CPU based on shared memory with multi-threading, cluster based on message passing, and their hybrid. In the multicore CPU architecture, the global parallelization is used, where a genetic loop, selection, crossover, mutation, and evaluation is performed in each thread. On the other hand, in the cluster architecture, the island model is used, where each computing node generates different initial population and executes the complete algorithm. After that, best solutions from slave nodes join the new population, and the new population is sent to slave nodes. The hybrid architecture combines them, in particular, each computing node has different population and it executes the complete algorithm while performing genetic loop with a number of threads.

The work of Shayeghi et al. [107] and that of Davis et al. [39] propose pool based Birmingham cluster genetic algorithm (Pool-BCGA), which is an extension of generation based BCGA [70]. The traditional generation based BCGA, which combines the local minimization with GA, is a sequential code where local optimization of individuals are not independent from one another. On the other hand, the proposed Pool-BCGA is a kind of asynchronous global parallelization model, in which one global population pool is raced from several independent GA runs. Each GA run performs genetic operators such as mutation and crossover and execute local minimization for newly generated solution. The cluster environment with up to 240 processors is used and each pool subprocess of the proposed method running on 48 processors.

The work of Roberge et al. [100] reports the application of PGA implemented on GPU to optimize switching angle of multilevel inverters. This paper employs the hybrid approach, the island model and the evaluation parallelization. The purpose of this parallelization is to enable real-time control of multilevel inverters. For the evaluation parallelization, four strategies are taken into account to parallelize and accelerate the fitness evaluation on GPU. The proposed implementation achieved a speedup of 469× and the execution times ranged from 38 to 164 ms allowing for real-time control.

Since many processors on the cluster architecture or GPU can be recently utilized, the global parallelization and the evaluation parallelization are suitable to such environment because of their scalability. For this reason, the hybrid of them with other parallel implementation, such as the island model, gathers much attention.

## 4.2 Hardware

Hardware is one of the most important component to implement PGA in practice. There are many kinds of hardware architecture to be employed to implement parallel systems. In Table 6, three hardware architectures are employed; multicore CPU, cluster and GPU.

**Multicore CPU:** Multicore CPU is simple but powerful hardware architecture, on which two or more processing cores are embedded on the CPU. Usually, multiple cores share one memory space (shared memory) and can exchange data through shared memory. In Table 6, seven works employ multicore CPU. When multicore CPU is used in PGAs, 3-8 processors are used for parallelization. From the point of view of the PGA model, five works out of seven employs the parallel island model, while two works employs the general parallelization.

**Cluster:** Cluster architecture is constructed with two or more PCs with singlecore or multicore CPU and distributed memories. PCs in cluster architecture are generally connected each other through local area network (LAN) and/or wide area network (WAN), and communicate by using message passing and/or socket communication. In cluster architecture, the number of processors can be increased by increasing the number of PCs, which is easier than multicore CPU architecture because it must increase the number of processors in one CPU chip. Actually, when cluster architecture is used in the PGA works in Table 6, up to 240 processors are utilized. The

work of Tosun et al. [120] utilizes an HC system with totally 368 processors, consists of 46 nodes, each with two CPUs and each CPU having four cores.

**GPU:** Graphical processing unit (GPU) is originally used to accelerate real-time image processing, while it has recently attracted attention for the use of general purpose, in particular, evolutionary computation and deep neural networks. Unlike multicore CPU, GPU has hundreds or thousands of processors in one chip. Three papers in Table 6 utilize graphical processing units (GPUs) with hundreds cores, each having 512 cores [100], 448 cores [88], and 384 cores [31]. GPUs are applied to an IPGA models, global parallelization, parallel island model, and parallel cellular model.

Apart from these hardwares, several hardware architectures can be taken into account to implement PGA, in particular, grid computing, cloud computing, mobile/smartphone, sensor network, and FPGA.

**Grid computing:** Grid computing is a technology that aggregates and uses resources such as distributed processors, data, storage systems, and networks, and make them available as one huge resource. One of the important aspects of grid computing is to gather surplus computing resources and make effective use of it in demand areas. In PGA domain, grid computing technology is attracted attention, for example, the work of Georgiev and Atanasov [55], and the one of Oliveira et al. [123] implement an island model PGA on grid computing environment.

**Cloud computing:** Cloud computing is a paradigm for delivering computing power, database storage, applications, and other IT resources through a cloud services platform via the Internet. Enterprise cloud services are provided such as Amazon Web Service, Google Cloud Platform, and Microsoft Azure. Such services are certainly utilized for the use of parallel computing including PGA executions, actually several works study implementation of PGAs on cloud system [47, 81, 103, 132].

**Mobile/Smartphone:** In the last decade, smartphones and mobile devices have rapidly become widespread, and their processing capability has remarkably improved. For this reason, many researchers study to make use of smartphones and mobile devices as computing resources like cluster or grid computing architecture [27].

**Sensor network:**

**FPGA:** Field programmable gate array (FPGA) is a reconfigurable integrated circuit after production. To reconfigure FPGA, hardware description language (HDL) is generally used. The main advantage of FPGA is its flexibility that enables high-efficiency and high-speed processing of specific applications by configuring logic circuit specified to tasks. Another advantage is its high parallel performance. By constructing same logic circuits for a certain process, it can be executed in parallel during one CPU clock. Utilizing these advantages, PGA architecture using FPGA has been proposed in several studies [52, 62].

## 4.3 Software

Several APIs are utilized to quickly and certainly implement PGA algorithms in practice. In Table 6, four APIs are employed: Parallel Computing Toolbox, Distributed Computing Server, MPI, and CUDA.

**Parallel Computing Toolbox:** Parallel Computing Toolbox is used in two papers [18, 99] with MATLAB. Parallel Computing Toolbox is provided by MathWorks and supports to solve computationally and data-intensive problems using multicore processors.

**Distributed Computing Server:** Distributed Computing Server is also provided by MathWorks to support parallel computing on distributed computing environment such as cluster computing architecture. Two papers [45, 87] employ Distributed Computing Server with MATLAB on cluster architecture.

**MPI:** Message passing interface (MPI) is a standard for message passing on parallel computing architecture. MPI libraries such as OpenMPI and MPJ are used as major parallel API with Fortran, C++, and Java. Actually, three works [76, 90, 120] in Table 6 employ MPI on cluster computing architecture.

Table 10. Academic and generated problem instances in the paper cited more than 20 times

| Ref. | Instance | Domain | Data set |
|---|---|---|---|
| Ishibuchi et al. [67] | Academic | Data mining | UCI machine learning repository [46] and KEEL project [16] |
| Osaba et al. [86] | Academic | Routing problem | TSPLIB [97] and the CVRP data set by [60] |
| Hong et al. [65] | Problem generation | Data mining | Randomly generated 10,000 purchase transactions with 64 items |
| Tosun et al. [120] | Academic | Quadratic Assignment Problem | QAPLIB [26] |
| Allahviranloo et al. [18] | Problem generation | Routing problem | Randomly generated fuzzy selective vehicle routing problem (FSVRP) consisting of 15 nodes and 2 vehicles and 200 nodes and 20 vehicles |
| Pinel et al. [88] | Problem generation | Task scheduling | Six different instance sizes, from 512 tasks×16 machines 65,536 tasks×2,048 machines, generated according to [17] |
| Dorronsoro et al. [48] | Problem generation | Task scheduling | 512 tasks with 16 machines and 2,048 tasks with 64 machines, with two heterogeneities, generated according to [17] |
| Kurdi [73] | Academic | Job shop scheduling | Standard JSSP benchmark instances by [20, 53, 74] |
| Cekmez et al. [31] | Academic | Routing problem | TSPLIB [97] |

**CUDA:** Especially for GPU architecture, CUDA toolkit is used. CUDA developed and provided by NVIDIA is a platform for general purpose parallel computing on GPU architecture, and C/C++ compiler and APIs are provided. On the PGA work employing GPU in Table 6, all three works [31, 88, 100] employ CUDA on GPU developed by NVIDIA.

In order to implement PGAs, several EA libraries are utilized to quickly put proposed algorithm in practice. In particular, a few papers in Table 6 employ libraries such as GEATbx, SiPESC.OPT, and jMetal.

**GEATbx:** The Genetic and Evolutionary Algorithm Toolbox (GEATbx) [89] is a toolbox for MATLAB. It does not only contain many optimization algorithms and problem instances and examples to run them, but also many helpful extensions like visualization of optimization process, multi-objective optimization, constraint handling, problem-specific initialization and visualization, and multi-strategy and multi-population support. In Table 6, the work of Devos et al. [45] utilizes GEATbx.

**SiPESC.OPT:** The work of Yang et al. [131] utilizes SiPESC.OPT [130] developed and provided by Dalian SiPESC CO., Ltd.

**jMetal:** jMetal framework [49] is implemented by Java language and includes single-objective, multi-objective and parallel algorithms, problem instances, quality indicators, and variable representations. jMetal is written as the object-oriented architecture so that it enables users to quickly develop their own algorithm and solve their own optimization problem by the state of the art EA algorithms. Dorronsoro et al. [48] utilize jMetal library in their work.

## 4.4 Academic and generated problem instances

This section shows the details of academic and generated problem instances used in PGA papers cited more than 20 times. Academic instances are very useful to compare the proposed PGA method with other previous ones because they can be compared with fair condition. On the other hand, generated problem instances are also useful because they can be easily scaled up their problem size. This feature is suitable to measure scalability or robustness of proposed PGA algorithm from the point of view of problem size.

Table 10 summarizes the characteristics of the instances of academic and problem generation. "Instance" column indicates corresponding paper employs academic instance or generated problem instance, which "Domain" column indicates the problem domain of instances used in papers. "Data set" column indicates where problem instances come from or how they are generated.

As benchmark problems, well-known NP-hard problems are employed in PGA domain. Routing problem and task scheduling are usually employed as test instances in PGA domain. The work of Osaba et al. [86] and that

Table 11. Applications in the papers cited more than 20 times

| Ref. | Application | # objectives | Fitness | Constraint | Solution |
|---|---|---|---|---|---|
| Roberge et al. [99] | Path finding | Mono | Static | No | Real-value |
| Liu et al. [76] | Traffic | Mono | Static | Yes | Real-value |
| Devos et al. [45] | Data mining | Mono | Static | No | Real-value (binary coded) |
| Porta et al. [90] | Land use planning | Mono | Static | No | Discrete |
| Sahingoz [101] | Path finding | Mono | Static | No | Discrete |
| Shayeghi et al. [107] | Nanoscience | Mono | Static | No | Real-value |
| Roberge et al. [100] | Electronics | Mono | Static | No | Real-value |
| Yang et al. [131] | Building structure | Mono | Static | No | Mixt[illegible]eal-value and discrete) |
| Park et al. [87] | Building structure | Multi | Static | Yes | Real-value |
| Davis et al. [39] | Nanoscience | Mono | Static | No | Real-value |

of Cekmez et al. [31] employ [illegible]IB [97] that is a standard library for TSP problem instance. The work of Allahviranloo et al. [18] applies PGA to solve fuzzy selective vehicle routing problem (FSVRP) and they employ randomly generated small and large routing [illegible]work for testing the proposed method. Other than TSP, the capacitated (CVRP) data set provided by NEO research group [60] is used in the work of Osaba et al.. Larger size of TSP is solved in the work of Honda et al. [63], in which 100,000-city scale TSPs, called Art TSP instances [6] that provide continuous-line drawings of well-known pieces of art, is solved with PGA with [illegible]ge assembly crossover (EAX).

Test instance[illegible] of task scheduling are randomly generated for given number of tasks and given number of machines. The[illegible] can be easily change their scale by changing the number of tasks and machines. In the work of Pinel et al. [88], six different instance sizes are tested, spe[illegible]y 512 tasks × 16 machines, 4, 096 [illegible], 8, 192 × 256, 16, 384 × 512, 32, 768 × 1, 024, and 65, 536 × 2, 048, while the work of Dorronsoro et al. [48] employs 512 tasks × 16 machines an[illegible]048 tasks × 64 machines with high and low heterogenities of the tasks and machines.

In the work of genetic based machine learning (GBML), UCI machine learning repository [46], which is a well-known collection of data set in the machine learning domain, is used. Quadric Assignment Problems (QAPs) and job shop schedulings

### 4.5 Applications

Many researchers apply PGA to solve problems in real-world application, in particular, path finding, traffic, data mining, land use planning, nanoscience, electronics, and building structure. In this section, we discuss on these application[illegible]ved in PGA papers cited more than 20 times.

Tab[illegible] shows th[illegible]mmary of appli[illegible]ons and their problem characteristics in the paper cited more than 20 times. "Application" column indicates a real-world application treated i[illegible] corresponding paper. "[illegible]bjectives" column indicates the number of objectives of the tar[illegible]em, while "Fitness" column indicates whether the fitness value of the target problem is sta[illegible] dynamic. "Constraint" co[illegible]n indicates whether the target problem contains some constraints or not, while "Solution" column indicates representation of solution. The details of studies related to real-world applications are discussed below.

**Path finding:** The work of Roberge et al. [99] uses PGA and parallel PSO to produces paths of unmanned aerial vehicles (UAVs). Each path is represented as a number of waypoints (3D coordinates) of UAV, and is evaluated by penalizing path length, average altitude, path through da[illegible] zones, required power of UAV, colliding with the ground, required fuel, and path that cannot be smoothed. Island model parall[illegible]tion is used, in which the migrations occur synchronously, ten times throughout the optimization process and new subpopulations are generated by rando[illegible]rmutation of a global population. The experiment is run on two quad-core CPUs (to[illegible]

[6] http://www.math.uwaterloo.ca/tsp/data/art/index.html

8 cores). To implement the algorithm, the authors employs MATLAB and Parallel Computing Toolbox. They achieved a quasi-linear speed up of 7.3 on 8 cores and an execution time of 10 s for both algorithm, which shows the possibility of real-time path planning for UAV on a standard multi-core CPU. The work of Sahingoz [101] also solves the UAV path finding problem on PGA. In this work, paths of multiple UAVs are optimized simultaneously. The global parallelization is employed and each processor of parallel computing nodes performs selection, crossover, mutation and local search. This work employs multicore CPU with four nodes and four different UAV paths are optimized. Although not listed in Table 6, some other researches tackle the UAV optimization. For example, Shaferman and Shima [106] applies coevolutional PGA to optimize a heterogeneous UAV team tracking a group of targets, while the work of Cekmez et al. [32] implements the multi-UAV optimization on the GPU architecture with CUDA.

**Traffic:** Liu et al. apply PGA to traffic management, in particular the practical speed-based toll design problem for the cordon-based Electronic Road Pricing (ERP) system in Singapore [76]. In this problem, toll fare pattern for each charging link is optimized to maximize the total social benefit and average travel speed. Each newly generated chromosome is evaluated based on a the probit-based stochastic user equilibrium problem with given toll charges, which needs much computational efforts due to the high computational cost of the Monte Carlo simulation in each iteration. The global parallelization is chosen to evaluate each solution in parallel, and when 30 processors are used, the computation is accelerated by nearly 11 times, where an execution time decreases from around 30 hours to 2.7 hours. Since traffic optimization requires much computational efforts to simulate decision making of drivers for traffic congestion, PGA is widely utilized to speedup the optimization [2, 91–93, 108].

**Data mining:** The work of Devos et al. [45] applies PGA to data pre-processing and model selection of SVM classification. In particular, the chromosome consists of three parts of coding the two meta-parameters of SVM, $C$ and $G$, and $p$ pre-processing to be applied to the data. The global implementation is employed and each slave decodes the chromosome, applies pre-processing, construct SVM model with decoded meta-parameters, and evaluates this model by $k$-fold cross-validation. They employ High Performance Computing Cluster (Transtec AG, Tübingen) consisting of one master node and four computing nodes using 2 dual core processors, in total 16 processors are available. The proposed method is applied to the classification task of olive oil coming from Liguria (a coastal region of north-western Italy) and olive oil samples from other regions of Italy on the basis of NIR or MIR spectra. The experimental result reveals that the SVM with pre-processing and parameter optimized by PGA achieves higher classification accuracy than the previous classification methods. A combination of PGA with data mining or machine learning is studied by other researchers, for example, feature selection [23, 50, 61, 109], generating fuzzy rule set [33, 114].

**Land use planning:** The work of Porta et al. [90] uses PGA for land use planning, in which each piece of land is allocated to the best category according to certain criteria and restrictions. They consider three implementations of PGA, the global, island, and their hybrid, on different architectures, multicore CPU based on shared memory with multi-threading, cluster based on message passing, and their hybrid. The work of Cao and Ye [30] also applies PGA, in particular course-grained (island) PGA, to land use planning. The full connected topology is used and PGA is implemented on multicore CPU with four processors. The proposed model is applied to the land use planning problem in Tongzhuo Newtown, Beijing, China.

**Nanoscience:** The work of Shayeghi et al. [107] applies Pool-BCGA to solve global geometry optimization of molecular cluster, in which local optimizations are the bottlenecks in a global optimization when using *ab initio* method. The cluster environment with up to 240 processors is used and each pool subprocess of the proposed method running on 48 processors. The proposed method performs a linear scale-up while the traditional BCGA plateaus when using a large number of cores. The work of Davis et al. [39] also applies PGA to an optimization in the nanoscience domain, in particular the global optimization of $Ir_N$ ($N$ = 10-20) clusters directly at the density functional theory level of theory. They use the Birmingham parallel genetic algorithm (BPGA), a new open-source utilizing Pool-BCGA proposed in [107]. The BPGA calculations are performed on the UK's national supercomputer

ARCHER in this paper. Each is run in parallel with eight instances of the code operating on the pool. The proposed method successfully finds the putative global minimum structures of $Ir_N$ ($N$ = 10-20) clusters. Their proposed PGA is applied to optimization of other nano particles, MgO(100)-supported AuIr sub-nanoalloys [38], RuâĹŠPt binary nanoalloys [43], and MgO(100)-supported Pd, Au and AuPd nanocluster structures [66].

**Electronics:** The work of Roberge et al. [100] reports the application of PGA implemented on GPU to optimize switching angle of multilevel inverters. The purpose of this parallelization is to enable real-time control of multilevel inverters. This paper employs the hybrid approach, the island model and the evaluation parallelization. For the evaluation parallelization, four strategies are taken into account to parallelize and accelerate the fitness evaluation on GPU. The proposed implementation achieved a speedup of 469× and the execution times ranged from 38 to 164 ms allowing for real-time control.

**Building structure:** The work of Yang et al. [131] applies the global parallelized NSGA-II to building energy consumption optimization, which uses the simulation model of the KUBIK building, locating at Bilbao Spain and developed in EnergyPlus, to evaluate fitness value. The experiment uses 48 computing nodes constructed of eight computers with six-core processor. The parallel optimization framework with 48 processors achieves the speedup ratio of 39.5. The work of Wang et al. [87] proposes distributed NSGA-II, a multi-objective genetic algorithm, to seismic retrofit design of 2D steel frame structure and 3D irregular reinforced concrete structure. Since a time-consuming nonlinear structural analysis is needed for the evaluations of seismic performances of a retrofitting design, they employ the global parallelization to evaluate solutions in parallel. They use the cluster architecture, in which 16 PCs with quad-core processors are connected through network switching, in total this cluster has 64 processors. Distributed Computing Server provided by MathWorks is used. It revealed that the distributed algorithm on a cluster of multicore PCs can reduce the computational time for the optimization without deteriorating the quality of the optimal solutions. In particular, although the efficiency decreases by increasing the number of processors, they achieve 75% efficiency with 64-core configuration.

## 4.6 Summary

In this section, we show the review of the literature in the recent five years by focusing on papers cited more than 20 times. Firstly we discussed on the point of view of PGA models, and then we discussed on two aspects, hardware and software used to implement proposed PGAs. Next, we described academic and generated problem instances used to demonstrate the effectiveness of proposed PGAs. And finally, we described real-world applications solved by the recent PGA papers. This section is summarized as follows:

- The island model and the global parallelization are two major implementation of PGAs, while the hybridization approach like the combination of the island model and the global parallelization, or the one of the global parallelization and the evaluation parallel are attracted much attention.
- The multicore CPU with several processors and the cluster architecture with tens or hundreds processors on many computing nodes are mainly employed. On the other hand, GPU with hundreds of cores recently employed especially by using CUDA library.
- To implement PGAs, several APIs like Parallel Computing Toolbox, Distributed Computing Server, MPI, and CUDA are utilized, while useful libraries help authors easily implement PGAs, for example, GEATbx, SiPESC.OPT, Paradiseo-metal.
- Academic and generated problem instances are utilized to test PGA methods, in particular they are useful to investigate the scalability of PGA methods because such academic and generated problem instances can be easily scaled up by increasing the dimension of design variables.
- Many researchers pay attention to real-world application of PGAs. They apply PGAs to several domains, like path finding, traffic, data mining, land planning, nanoscience, electronics, and building structure. Many of them are computationally expensive, or have to be solved in short computing time.

## 5 CHALLENGES AND NEW TRENDS

This section contains a brief summary of open challenges for the further development of PGAs, what could be used as a set of research lines to foster in new master and PhD theses. Also, we include information on already developing technologies and fields related to PGAs, what could be used to guide interested readers to domains new to them. We then make a summary of what is worth to be addressed and what is worth to know because it is happening as of today, hopefully making here a case of discussion and thinking for the PGA community.

Again, this is a reflection based on the knowledge of authors of this article, and then a topic for open debate, of course. Indeed, we will merge callenges and future trends in the next list of items, since they both interplay in complex manners. Let us start.

**Few seconds execution:** In real time applications like in critical systems (Industry 4.0, drivers, cars, planes...) a fast execution is mandatory. PGAs running completely withing a few seconds (or milliseconds) is a challenge for present research. For this, we should probably need more than a fast parallel hardware, and then we should go for theory (to reduce the number of function evaluations) and especialized operators.

**Beating the state-of-the-art:** PGAs are often just used to reduce the running time, and authors base their contribution in this reduction. But PGAs allow a much larger power than this, and they could ease the reduction of number of fitness function calls and produce and increment in the precision of the computed solution as well. New PGA models and implementation are wanted achieving this *parallel* maximization of benefits and reduction of costs.

**Scalability:** One main concern in present research is how narrow the studies published in impact journals. In particular, authors often only address one problem instance, or a few ones that share a similar dimensionality. A main challenge is to face PGAs to many problem instances where all of them are of very different dimensions. For example, go for thousands and millions of variables when solving SAT, or for hundreds, thousands and millions of cities when solving TSP [41, 58]. If the proposed algorithm shows a quasi-linear scalability (growth of the effort of the same algorithm when facing increasingly large problems), then this technique is worth to publish and learn [15]. The number of new techniques is so large nowadays, and the need for scalability is so normal in the real world, that we should only care on algorithms showing with numerical data that they scale appropriately.

**Robustness:** This is an often mentioned issue in most search, optimization, and learning studies at present, though not so often well defined. We can understand robustness as the *persistance of good results* shown by an algorithm under changes (of different intensity) in its parameters [64, 128]. Having PGAs that show robustness in this sense will make then reliable tools for final applications in the real world. The feature of metaheuristics consisting in giving a different answer to the same problem in different runs (non-determinism) sometimes is not welcomed by experts, who want a reliable and even same result always. At least considering probability of distributions instead of scalars (e.g. on TSP, consider that travel times are not numbers, but normal distributions with an average and a standard deviation) and reporting on confiden intervals for algorithms is a demanded feature of modern solvers that researchers should be aware of [8, 37]. Robustness is often also understood as algorithms dealing with uncertainties in the data, or even dealing with dynamic environments [11], where the same individual can get several different fitness values along the search done in the algorithm, clearly defining new topics for research.

**Multiobjective:** The combination of PGAs and MO studies/concepts is a classic now in literature [71, 78]. The checking of constraints in complex MO problems take a lot of time, and PGAs are a way to go there. In many MO problems, checking constraints require more time than evaluating the different objective functions. In other cases, computing hypervolume can be eased by using a PGA. Still in other situations, new models for solving MO problems can be built my decentralizing the MO algorithm, thus having more efficient algorithms that could construct better Pareto fronts in diversity and quality. Also, computing more solutions

in the front can come as a benefit of using a PGA. Finally, as to multicriteria decision making (MCDM) we still have to see interactive PGAs dealing with one or more users stating their preferences, either offline or online, in an interactive way so as to allow new applications in design (e.g. designing a building, creating a painting...) and market (e.g. analyzing bank data, determining the best medical treatment for a patient...).

**Interactive/online:** This topic is still to rise, but it is clear that, if users need a real time answer to their queries for an optimal solution to a complex real problem, then PGAs are the way to go [85]. First, modern hardware would be needed (GPUs mainly, but also FPGAs or cloud computing), and thus the importance of building algorithms for new platforms. Second, graphical user interfaces (and vocal ones, and others) would have a say here. We also link now to very efficient execution (seconds or milliseconds) and user preferences, targeting a new body of services where a PGA is in the heart of the intelligence offered to the user. These systems sometimes need to wait for the user, and then go for a quick answer to the defined problem, in a kind of burst behavior that open new avenues on how all this should be managed.

**Body of knowledge:** The difficulty for having an explicit body of knowledge in PGAs is a main problem for its development as a research field. Researchers have a hard time in finding relevant papers, there are no clear benchmarks to evaluate the quality of PGAs, and the list of best practices is still weak and splitted in different journals and conferences. Often, researchers interested in PGAs only read and publish in journals on GAs, but not on parallelism. The contrary also happens, since there is no such a journal or conference with yearly execution where to find data on this domain. In order to know more on the body of knowledge, researchers should at least consult the previous surveys on this topic [3, 4, 10, 14, 28, 69, 72, 116, 119, 122]. We hope to make a contribution to this body of knowledge with this article, but obviously it should happen at the level of the interested community, and community is still sparse, since most researchers are going for applications of PGAs, and not for their fundamentals. That is a must; going for the working principles and bases for the construction and analysis of PGAS themselves is a real future challenge.

**Trade-off usability/efficiency:** PGAs are often used because researchers want to build a complex solver that is efficient at the same time. Sometimes, making the algorithm usable means to forget on the internals of the algorithm and focus on the user interface, full of data with lots of data relevant for the application domain. And thus, it is normal that efficiency and even science are sacrified for the sake of the nice application at hands. We here warn that PGAs, and metaheuristics in general, are not magic recipes for solving any problem. They are not black boxes amenable for users to build their applications while forgetting on how the algorithm is working. It is very normal that this vision make researchers happy at first when using PGAs but sooner later a wall is faced, where no advances are possible without going into the algorithm operations. In particular, efficiency is a must, even if we talk of a few seconds of execution, since in this domain we care also on small times, always having scalability in mind, and always wanting an analysis of the algorithm to be able of further enhancements [29, 75, 95, 124]. Not being able of this tradeoff is the cause for stagnation in research, and for building new software parallel tools that easily go inefficient on larger problems just because researchers do not want (have no time) to enter the guts of the PGA used.

**Big data+parallel:** Although no one has a clear idea of what big data is, the fancy term got known after several initial attemps to use parallel platforms without caring of parallel details. Big data researchers use software like Hadoop and Spark as an abstract layer to use parallelism to harvest gigabytes of information to offer a statistical or intelligent conclusion. Since map/reduce is the main paradigm here [29, 113, 129], found in MPI and other parallel software and systems previous to the existence of big data, much can be said on how to configure and program on big data to efficiently use the underlying computing services. If all big data applications use Spark (or similar), and if Spark (and similar) software tools are a way of using map/reduce parallelism (to forget on the internal parlalel platform) then big data is actually concerned by parallel issues, and algorithms, and metrics, and body of knowledge. So we encourage researchers interested in big data to first understand the kind of algorithms that can be run in parallel in a large amount of computers [5], so

that their solutions actually exploit the hardware and can go for big data, bit volatility big variability, and the rest of 'v' and in this domain [96].

**Edge/Fog computing:** A fact in research is that we proceed in waves of interest. And a new wave of interest is starting on not going centralized (as big data implies) for the analysis of data. Fog (or edge) computing [1, 19, 94, 98] claims that the computing should be done in the edge, that is, in the devices near the input of data to the system. This means running algorithms on smart phones, routers, and some other intermediate devices, no longer using a set of computers as the central location for data processing and analysis. Then building new PGAs running on portable devices (tablets, smartphones), small computing limited devices (Raspberry Pi 3, Arduino, wearables) and other edge devices (routers, like the ones from Cisco running iOX) is a major challenge and a future trend that we will see to happen in the coming years. The ones arriving quickly there will be well positioned for the innovation, research and development of the next generation of algorithms and services.

**Small devices:** After making clear that it is important to open new lines of research concerning small devices, one wonders on what are the challenges and interesting venues here. First, we can mention the obvious need to run complex algorithms with a slow set of cores (compared to desktop machines), a short memory, and with a low impact in battery consumption [9, 36, 77, 82]. Algorithms should then focus in theoretical results to perform an efficient search [35], but also new PGA models could arise form using many cores in a small device and linking the algorithm between different devices at the same time. Second, we need algorithms aware of the hardware [84, 102], and this means they perform operations that change in time depending on the availability of computing resources; this is still not well known: how mutation, crossover, local search should be implemented to perform a meaningful numerical search while at the same time we adapt to a volatile set of small devices, running/communication errors, and the computing resources that get exhausted along the life of the algorithm. How to build algorithms here, so as to deal with thousands of devices, and to address problems in a crowd (voluntary) kind of computing is a wide and interesting line of research that will make PGAs and AI ubiquitous.

**Exascale computing:** This refers to computing systems capable of at least one exaFLOPS, or a billion billion calculations per second ($10^{18}$ floating point operations per second or FLOPS) [21, 25, 40]. This is the gold target for High Performance Computing (HPC) in the near future, where all computing labs in the world want to arrive. With such a power and parallel platforms, new unseen PGAs can be engineered and exploited for very complex problems. But also, this inmense computational platform can be the target of studies that use PGAs. For example, a typical problem in large HPC platforms is that you cannot use a sensible or even a small set of all the computing units: arriving to 10% or 20% of all them is usually the barrier: how to use more of this power at the same time for a PGA? How to help schedulers to improve their performance by using PGAs for the decision making on the best execution plan for tasks? How to deal with PGAs where any packet delivery could potentially fail, or any component of the PGA could be removed from execution while still searching for a solution to our problem? An exciting set of possibilities then open in merging PGAs, their design, their application, and an exascale computing platform all together.

**Web services:** In most research domains linked to optimization, search, and learning, researchers offer numerical results, data files with benchmarks, documents and open software in form of libraries. However, there are more ways to use the knowledge we all build when researching, and building web services is one of them [104]. Indeed, an interesting one, since we could offer new optimization services for users through web services that are internally running PGAs on a local network (or multicore) [12]. That would be well accepted by practitioners, who will submit their problems (through a given description language, or XML file, or just source coude in Java) and wait for an answer. Building microservices and merging them into more complex solutions would be of interest in the future, specially for a cohort of researchers who want to use well tested and validated algorithms without going into their internals. Indeed, this could move

to a new kind of market of specialized solvers used through reliable web services, thus representing an interesting shared domain for researchers and companies.

## 6 CONCLUSIONS

We now end this article by summarizing the main contents and findings in it. Let us first remind of the intention of not only having a source for modern research in PGAs, but in including added value in every section so that it is actually useful to a wide audience.

On the contents, we tried to visit all common places, and then talked on models for PGAs, existing fresh works on this domain, and a guiding tour around the venues (conferences and journals) and places where PGAs are being endorsed. On conferences, ACM GECCO and IEEE CEC are preferred places for publishing in PGAs. On journals, the two first ones have a clear profile in pararellism, an explicit lesson that the "P" in PGAs has a deep importance.

As a global conclusion, it is clear that the domain is healthy, though there is a trend in focusing in the applications and not that much in the algorithms themselves. This is good in the short term but not so good in the long term if we want to understand and better exploit these techniques for future practice. We really encourage researchers to go on the fundamentals and understand them well before passing to focus in the application itself, so as to avoid the classic "wall" of no further enhacements in the performance for a lack of knowledge on the solver used.

The environment in research on PGAs is shifting in topics and places also. China is emerging as a source of many new works, and our understanding of PGAs would need to master the taxonomy on hardware and software that is somewhat complex because of the many advances in these two domains. Researchers and practitioners should pay attention to the fundamentals of the models, the best practices in non-deterministic algorithms, and the basic concepts existing in the domain of parallelism and HPC that are, to some degree, continuously missed in present works where the application or at most the algorithm is the target, while a whole body of information exists for parallel processing that could lead this domain to new levels of creative thinking and highly efficient solvers.

We hope that our tables and figures represent a quick means to understand this domain. We also tried to contribute in giving details as much as possible but without getting lost because of them. A global picture of the domain is a nice lesson that this survey can help grasping. Indeed, we went into details on new topics and open challenges so that new PhD theses can have a starting point based in the many suggestions implicit in this paper when a reader look in a combination of them at all the sections.

This survey represents a big database of recent work, and also a long list of suggestions and data and tools that will hopefully help others in making, not only research, but development and even innovation to transfer efficient solvers to companies. Of course, more work still has to be done in defining the fundamentals of PGAs, and in developing scientific thinking in the future advances in this field. We do believe PGAs are precious tools what could help anyone out there to improve his/her techniques, thus solving open problems and even leading to funny and intuitive ways of addressing complex problems.

## ACKNOWLEDGMENTS

This research has been partially funded by the Spanish MINECO and FEDER projects TIN2016-81766-REDT (http://cirti.es) and TIN2017-88213-R (http://6city.lcc.uma.es).

## REFERENCES

[1] 2015. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are.* White paper. Cisco Sytems, Inc., San Jose, CA.

[2] Ghassan Abu-Lebdeh, Hui Chen, and Mohammad Ghanim. 2016. Improving Performance of Genetic Algorithms for Transportation Systems: Case of Parallel Genetic Algorithms. *Journal of Infrastructure Systems* 22, 4 (2016), A4014002. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000206

[3] Panagiotis Adamidis. 1994. *Parallel evolutionary algorithms: A review.* Technical Report. University of Thessaloniki.

[4] Enrique Alba. 2005. *Parallel Metaheuristics: A New Class of Algorithms.* Wiley.

[5] Enrique Alba. 2005. *Parallel metaheuristics: a new class of algorithms.* Vol. 47. John Wiley & Sons.

[6] Enrique Alba, Christian Blum, Pedro Isasi, Coromoto Len, and Juan Antonio Gmez (Eds.). 2009. *Optimization Techniques for Solving Complex Problems.* John Wiley & Sons, Inc. https://doi.org/10.1002/9780470411353

[7] Enrique Alba, Christian Blum, Pedro Isasi, Coromoto León, and Juan Antonio Gómez (Eds.). 2009. *Optimization Techniques for Solving Complex Problems.* Wiley.

[8] Enrique Alba, Francisco Chicano, and Gabriel Luque (Eds.). 2017. *Smart Cities - Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings.* Lecture Notes in Computer Science, Vol. 10268. Springer. https://doi.org/10.1007/978-3-319-59513-9

[9] Enrique Alba, Francisco Chicano, and Gabriel Luque (Eds.). 2017. *Smart Cities - Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings.* Lecture Notes in Computer Science, Vol. 10268. Springer. https://doi.org/10.1007/978-3-319-59513-9

[10] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. 2013. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research* 20, 1 (2013), 1–48. https://doi.org/10.1111/j.1475-3995.2012.00862.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2012.00862.x

[11] Enrique Alba, Amir Nakib, and Patrick Siarry. 2013. *Metaheuristics for dynamic optimization.* Vol. 433. Springer.

[12] E. Alba and J. G. Nieto. 2005. *ROS (Remote Optimization Service).* Technical Report Informe Técnico ITI 05-08, Dpto. de Lenguajes y CC.CC. Universidad de Málaga. (in Spanish).

[13] E. Alba and M. Tomassini. 2002. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 6, 5 (Oct 2002), 443–462. https://doi.org/10.1109/TEVC.2002.800880

[14] Enrique Alba and José M. Troya. 1999. A Survey of Parallel Distributed Genetic Algorithms. *Complex.* 4, 4 (March 1999), 31–52. https://doi.org/10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.3.CO;2-W

[15] Enrique Alba and José M. Troya. 2002. Improving flexibility and efficiency by adding parallelism to genetic algorithms. *Statistics and Computing* 12, 2 (01 Apr 2002), 91–114. https://doi.org/10.1023/A:1014803900897

[16] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. 2009. KEEL: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing* 13, 3 (01 Feb 2009), 307–318. https://doi.org/10.1007/s00500-008-0323-y

[17] Shoukat Ali, Howard Jay Siegel, Muthucumaru Maheswaran, Debra Hensgen, and Sahra Sedigh. 2000. Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems. *Tamkang Journal of Science and Engineering* 3, 3 (2000), 195–207.

[18] Mahdieh Allahviranloo, Joseph Y.J. Chow, and Will W. Recker. 2014. Selective vehicle routing problems under uncertainty without recourse. *Transportation Research Part E: Logistics and Transportation Review* 62 (2014), 68 – 88. https://doi.org/10.1016/j.tre.2013.12.004

[19] Muhammad Rizwan Anawar, Shangguang Wang, Muhammad Azam Zia, Ahmer Khan Jadoon, Umair Akram, and Salman Raza. 2018. Fog Computing: An Overview of Big IoT Data Analytics. *Wireless Communications and Mobile Computing* 2018 (2018), 22 pages.

[20] David Applegate and William Cook. 1991. A Computational Study of the Job-Shop Scheduling Problem. *ORSA Journal on Computing* 3, 2 (1991), 149–156. https://doi.org/10.1287/ijoc.3.2.149 arXiv:https://doi.org/10.1287/ijoc.3.2.149

[21] Muhammad Usman Ashraf, Fathy Alburaei Eassa, Aiiad Ahmad Albeshri, and Abdullah Algarni. 2018. Toward Exascale Computing Systems: An Energy Efficient Massive Parallel Computational Model. *International Journal of Advanced Computer Science and Applications* 9, 2 (2018). https://doi.org/10.14569/IJACSA.2018.090217

[22] Richard J. Bauer. 1994. *Genetic Algorithms and Investment Strategies.* Wiley.

[23] A. A. Besalatpour, S. Ayoubi, M. A. Hajabbasi, A. Yousefian Jazi, and A. Gharipour. 2014. Feature Selection Using Parallel Genetic Algorithm for the Prediction of Geometric Mean Diameter of Soil Aggregates by Machine Learning Methods. *Arid Land Research and Management* 28, 4 (2014), 383–394. https://doi.org/10.1080/15324982.2013.871599 arXiv:https://doi.org/10.1080/15324982.2013.871599

[24] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM COMPUTING SURVEYS* (2003), 268–308.

[25] Katherine Bourzac. 2017. Supercomputing poised for a massive speed boost. *Nature* 551 (11 2017), 554–556.

[26] Rainer E. Burkard, Stefan E. Karisch, and Franz Rendl. 1997. QAPLIB – A Quadratic Assignment Problem Library. *Journal of Global Optimization* 10, 4 (01 Jun 1997), 391–403. https://doi.org/10.1023/A:1008293323270

[27] F. Büsching, S. Schildt, and L. Wolf. 2012. DroidCluster: Towards Smartphone Cluster Computing – The Streets are Paved with Potential Computer Clusters. In *2012 32nd International Conference on Distributed Computing Systems Workshops.* 114–117. https://doi.org/10.1109/ICDCSW.2012.59

[28] Erick Cantú-Paz. 1998. A Survey of Parallel Genetic Algorithms. *CALCULATEURS PARALLELES* 10 (1998).

[29] Erick Cantú-Paz and David E Goldberg. 1999. On the scalability of parallel genetic algorithms. *Evolutionary computation* 7, 4 (1999), 429–449.

[30] Kai Cao and Xinyue Ye. 2013. Coarse-grained parallel genetic algorithm applied to a vector based land use allocation optimization problem: the case study of Tongzhou Newtown, Beijing, China. *Stochastic Environmental Research and Risk Assessment* 27, 5 (01 Jul 2013), 1133–1142. https://doi.org/10.1007/s00477-012-0649-y

[31] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz. 2013. Adapting the GA approach to solve Traveling Salesman Problems on CUDA architecture. In *2013 IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI)*. 423–428. https://doi.org/10.1109/CINTI.2013.6705234

[32] Ugur Cekmez, Mustafa Ozsiginan, and Ozgur Koray Sahingoz. 2016. Multi-UAV Path Planning with Parallel Genetic Algorithms on CUDA Architecture. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion)*. ACM, New York, NY, USA, 1079–1086. https://doi.org/10.1145/2908961.2931679

[33] Mojtaba Asadollahpour Chamazi, Behrouz Minaei Bidgoli, and Mahdi Nasiri. 2013. Deriving support threshold values and membership functions using the multiple-level cluster-based master–slave IFG approach. *Soft Computing* 17, 7 (01 Jul 2013), 1227–1239. https://doi.org/10.1007/s00500-012-0973-7

[34] Hossein Rajabalipour Cheshmehgaz, Habibollah Haron, and Abdollah Sharifi. 2015. The review of multiple evolutionary searches and multi-objective evolutionary algorithms. *Artificial Intelligence Review* 43, 3 (01 Mar 2015), 311–343. https://doi.org/10.1007/s10462-012-9378-3

[35] Francisco Chicano, L. Darrell Whitley, and Enrique Alba. 2011. A Methodology to Find the Elementary Landscape Decomposition of Combinatorial Optimization Problems. *Evolutionary Computation* 19, 4 (2011), 597–637. https://doi.org/10.1162/EVCO_a_00039

[36] Christian Cintrano and Enrique Alba. 2016. Genetic Algorithms Running into Portable Devices: A First Approach. In *Advances in Artificial Intelligence - 17th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2016, Salamanca, Spain, September 14-16, 2016. Proceedings*. 383–393. https://doi.org/10.1007/978-3-319-44636-3_36

[37] Christian Cintrano, Francisco Chicano, and Enrique Alba. 2017. Robust Bi-objective Shortest Path Problem in Real Road Networks. In *Smart Cities - Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings*. 128–136. https://doi.org/10.1007/978-3-319-59513-9_13

[38] Jack B. A. Davis, Sarah L. Horswell, and Roy L. Johnston. 2016. Application of a Parallel Genetic Algorithm to the Global Optimization of Gas-Phase and Supported Gold–Iridium Sub-Nanoalloys. *The Journal of Physical Chemistry C* 120, 7 (02 2016), 3759–3765. https://doi.org/10.1021/acs.jpcc.5b10226

[39] Jack B. A. Davis, Armin Shayeghi, Sarah L. Horswell, and Roy L. Johnston. 2015. The Birmingham parallel genetic algorithm and its application to the direct DFT global optimisation of IrN (N = 10-20) clusters. *Nanoscale* 7 (2015), 14032–14038. Issue 33. https://doi.org/10.1039/C5NR03774C

[40] Koen De Bosschere. 2012. *Applications, Tools and Techniques on the Road to Exascale Computing*. Vol. 22. IOS press.

[41] Kalyanmoy Deb and Christie Myburgh. 2016. Breaking the Billion-Variable Barrier in Real-World Optimization Using a Customized Evolutionary Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*. ACM, New York, NY, USA, 653–660. https://doi.org/10.1145/2908812.2908952

[42] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (Apr 2002), 182–197. https://doi.org/10.1109/4235.996017

[43] Ilker Demiroglu, Kezi Yao, Heider A Hussein, and Roy L. Johnston. 2017. DFT Global Optimization of Gas-Phase Subnanometer Ru–Pt Clusters. *The Journal of Physical Chemistry C* 121, 20 (05 2017), 10773–10780. https://doi.org/10.1021/acs.jpcc.6b11329

[44] James Devillers (Ed.). 1996. . Academic Press, London. https://doi.org/10.1016/B978-012213810-2/50005-0

[45] Olivier Devos, Gerard Downey, and Ludovic Duponchel. 2014. Simultaneous data pre-processing and SVM classification model selection based on a parallel genetic algorithm applied to spectroscopic data of olive oils. *Food Chemistry* 148 (2014), 124 – 130. https://doi.org/10.1016/j.foodchem.2013.10.020

[46] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[47] Yu-Shuang Dong, Gao-Chao Xu, and Xiao-Dong Fu. 2014. A Distributed Parallel Genetic Algorithm of Placement Strategy for Virtual Machines Deployment on Cloud Platform. *The Scientific World Journal* 2014 (2014), 12 pages.

[48] Bernabé Dorronsoro, Grégoire Danoy, Antonio J. Nebro, and Pascal Bouvry. 2013. Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution. *Computers & Operations Research* 40, 6 (2013), 1552 – 1563. https://doi.org/10.1016/j.cor.2011.11.014 Emergent Nature Inspired Algorithms for Multi-Objective Optimization.

[49] Juan J. Durillo and Antonio J. Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42 (2011), 760–771. https://doi.org/DOI:10.1016/j.advengsoft.2011.05.014

[50] El-Sayed M. El-Alfy and Mashaan A. Alshammari. 2016. Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algorithm in MapReduce. *Simulation Modelling Practice and Theory* 64 (2016), 18 – 29. https://doi.org/10.1016/j.simpat.2016.01.010 Advances on Information and Communication Systems.

[51] W. Fan, Y. Liu, B. Tang, F. Wu, and H. Zhang. 2016. Exploiting Joint Computation Offloading and Data Caching to Enhance Mobile Terminal Performance. In *2016 IEEE Globecom Workshops (GC Wkshps)*. 1–6. https://doi.org/10.1109/GLOCOMW.2016.7848902

[52] Rasoul Faraji and Hamid Reza Naji. 2014. An efficient crossover architecture for hardware parallel implementation of genetic algorithm. *Neurocomputing* 128 (2014), 316 – 327. https://doi.org/10.1016/j.neucom.2013.08.035

[53] H. Fisher and G.L. Thompson. 1963. Probabilistic learning combinations of local job-shop scheduling rules. In *Industrial Scheduling*, J.F. Muth and G.L. Thompson (Eds.). Prentice Hall, 225–251.

[54] Pablo García-Sánchez, Gustavo Romero, Jesús González, Antonio Miguel Mora, Maribel García Arenas, Pedro Ángel Castillo Valdivieso, Carlos M. Fernandes, and Juan Julián Merelo Guervós. 2016. Studying the effect of population size in distributed evolutionary algorithms on heterogeneous clusters. *Appl. Soft Comput.* 38 (2016), 530–547. https://doi.org/10.1016/j.asoc.2015.09.052

[55] D. Georgiev and E. Atanassov. 2014. Extensible framework for execution of distributed genetic algorithms on grid clusters. In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 301–306. https://doi.org/10.1109/MIPRO.2014.6859581

[56] Ali Ghaheri, Saeed Shoar, Mohammad Naderan, and Sayed Shahabuddin Hoseini. 2015. The Applications of Genetic Algorithms in Medicine. *Oman Medical Journal* 30, 6 (11 2015), 406–416. https://doi.org/10.5001/omj.2015.82

[57] M. Giacobini, M. Tomassini, A. G. B. Tettamanzi, and E. Alba. 2005. Selection intensity in cellular evolutionary algorithms for regular lattices. *IEEE Transactions on Evolutionary Computation* 9, 5 (Oct 2005), 489–505. https://doi.org/10.1109/TEVC.2005.850298

[58] David E. Goldberg, Kumara Sastry, and Xavier LlorÃă. 2007. Toward routine billion-variable optimization using genetic algorithms. *Complexity* 12, 3 (2007), 27–29. https://doi.org/10.1002/cplx.20168 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cplx.20168

[59] Y. Gong and A. Fukunaga. 2011. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *2011 IEEE Congress of Evolutionary Computation (CEC)*. 820–827. https://doi.org/10.1109/CEC.2011.5949703

[60] NEO Research Group. 2013. Vehicle Routing Problem. http://http://neo.lcc.uma.es/vrp/

[61] Alberto Guillén, Maribel García Arenas, Mark van Heeswijk, Dusan Sovilj, Amaury Lendasse, Luis Javier Herrera, Héctor Pomares, and Ignacio Rojas. 2014. Fast Feature Selection in a GPU Cluster Using the Delta Test. *Entropy* 16 (2014), 854–869.

[62] L. Guo, D. B. Thomas, Ce Guo, and W. Luk. 2014. Automated framework for FPGA-based parallel genetic algorithms. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*. 1–7. https://doi.org/10.1109/FPL.2014.6927501

[63] K. Honda, Y. Nagata, and I. Ono. 2013. A parallel genetic algorithm with edge assembly crossover for 100,000-city scale TSPs. In *2013 IEEE Congress on Evolutionary Computation*. 1278–1285. https://doi.org/10.1109/CEC.2013.6557712

[64] Chung-Chien Hong. 2014. A Robust Method for Designing the Parameters of Genetic Algorithm. *International Journal of Operation Research* 11 (2014), 051–063.

[65] Tzung-Pei Hong, Yeong-Chyi Lee, and Min-Thai Wu. 2014. An effective parallel approach for genetic-fuzzy data mining. *Expert Systems with Applications* 41, 2 (2014), 655 – 662. https://doi.org/10.1016/j.eswa.2013.07.090

[66] Heider A. Hussein, Jack B. A. Davis, and Roy L. Johnston. 2016. DFT global optimisation of gas-phase and MgO-supported sub-nanometre AuPd clusters. *Phys. Chem. Chem. Phys.* 18 (2016), 26133–26143. Issue 37. https://doi.org/10.1039/C6CP03958H

[67] H. Ishibuchi, S. Mihara, and Y. Nojima. 2013. Parallel Distributed Hybrid Fuzzy GBML Models With Rule Set Migration and Training Data Rotation. *IEEE Transactions on Fuzzy Systems* 21, 2 (April 2013), 355–368. https://doi.org/10.1109/TFUZZ.2012.2215331

[68] Nebro Antonio J., Durillo Juan J., Luna Francisco, Dorronsoro BernabÃĺ, and Alba Enrique. 2009. MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems* 24, 7 (2009), 726–746. https://doi.org/10.1002/int.20358 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.20358

[69] Fauzi Mohd Johar, Farah Ayuni Azmin, Mohamad Kadim Suaidi, A. S. Shibghatullah, Badrul Hisham Ahmad, Siti Nadzirah Salleh, Mohamad Zoinol Abidin Abd Aziz, and M. M. Shukor. 2013. A review of Genetic Algorithms and Parallel Genetic Algorithms on Graphics Processing Unit (GPU). In *2013 IEEE International Conference on Control System, Computing and Engineering*. 264–269. https://doi.org/10.1109/ICCSCE.2013.6719971

[70] Roy L. Johnston. 2003. Evolving better nanoparticles: Genetic algorithms for optimising cluster geometries. *Dalton Trans.* (2003), 4193–4207. Issue 22. https://doi.org/10.1039/B305686D

[71] Janusz Kacprzyk and Witold Pedrycz (Eds.). 2015. *Springer Handbook of Computational Intelligence*. Springer. https://doi.org/10.1007/978-3-662-43505-2

[72] D. S. Knysh and V. M. Kureichik. 2010. Parallel genetic algorithms: a survey and problem state of the art. *Journal of Computer and Systems Sciences International* 49, 4 (01 Aug 2010), 579–589. https://doi.org/10.1134/S1064230710040088

[73] Mohamed Kurdi. 2016. An effective new island model genetic algorithm for job shop scheduling problem. *Computers & Operations Research* 67 (2016), 132 – 142. https://doi.org/10.1016/j.cor.2015.10.005

[74] S. Lawrence. 1984. *Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. Graduate School of Industrial Administration, Carnegie-Mellon University.

[75] Yan Y. Liu and Shaowen Wang. 2015. A scalable parallel genetic algorithm for the Generalized Assignment Problem. *Parallel Comput.* 46 (2015), 98 – 119. https://doi.org/10.1016/j.parco.2014.04.008

[76] Zhiyuan Liu, Qiang Meng, and Shuaian Wang. 2013. Speed-based toll design for cordon-based congestion pricing scheme. *Transportation Research Part C: Emerging Technologies* 31 (2013), 83 – 98. https://doi.org/10.1016/j.trc.2013.02.012

[77] Oscar Luaces, José A. Gámez, Edurne Barrenechea, Alicia Troncoso, Mikel Galar, Héctor Quintián, and Emilio Corchado (Eds.). 2016. *Advances in Artificial Intelligence - 17th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2016, Salamanca, Spain, September 14-16, 2016. Proceedings*. Lecture Notes in Computer Science, Vol. 9868. Springer. https://doi.org/10.1007/978-3-319-44636-3

[78] Francisco Luna and Enrique Alba. 2015. Parallel Multiobjective Evolutionary Algorithms. In *Springer Handbook of Computational Intelligence*. 1017–1031. https://doi.org/10.1007/978-3-662-43505-2_50

[79] M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21, 2 (1979), 239–245. http://www.jstor.org/stable/1268522

[80] Jean-Yves Potvin Michel Gendreau (Ed.). 2010. *Handbook of Metaheuristics*. Springer US.

[81] Fereydoun Farrahi Moghaddam, Reza Farrahi Moghaddam, and Mohamed Cheriet. 2015. Carbon-aware distributed cloud: multi-level grouping genetic algorithm. *Cluster Computing* 18, 1 (01 Mar 2015), 477–491. https://doi.org/10.1007/s10586-014-0359-y

[82] J. A. Morell and Enrique Alba. 2017. Distributed Genetic Algorithms on Portable Devices for Smart Cities. In *Smart Cities - Second International Conference, Smart-CT 2017, Málaga, Spain, June 14-16, 2017, Proceedings*. 51–62. https://doi.org/10.1007/978-3-319-59513-9_6

[83] Amir Nakib and El-Ghazali Talbi (Eds.). 2017. *Metaheuristics for Medicine and Biology*. Studies in Computational Intelligence, Vol. 704. https://doi.org/10.1007/978-3-662-54428-0

[84] Ngoc Thanh Nguyen, Ryszard Kowalczyk, and Fatos Xhafa (Eds.). 2015. *Transactions on Computational Collective Intelligence XIX*. Lecture Notes in Computer Science, Vol. 9380. Springer. https://doi.org/10.1007/978-3-662-49017-4

[85] Kota Nomura and Makoto Fukumoto. 2018. Music Melodies Suited to Multiple Users' Feelings Composed by Asynchronous Distributed Interactive Genetic Algorithm. *International Journal of Software Innovation (IJSI)* 6, 2 (2018), 26–36.

[86] E. Osaba, F. Diaz, and E. Onieva. 2014. Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. *Applied Intelligence* 41, 1 (01 Jul 2014), 145–166. https://doi.org/10.1007/s10489-013-0512-y

[87] Keunhyoung Park, Byung Kwan Oh, Hyo Seon Park, and Se Woon Choi. 2015. GA-Based Multi-Objective Optimization for Retrofit Design on a Multi-Core PC Cluster. *Computer-Aided Civil and Infrastructure Engineering* 30, 12 (2015), 965–980. https://doi.org/10.1111/mice.12176 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12176

[88] Frédéric Pinel, Bernabé Dorronsoro, and Pascal Bouvry. 2013. Solving very large instances of the scheduling of independent tasks problem on the GPU. *J. Parallel and Distrib. Comput.* 73, 1 (2013), 101 – 110. https://doi.org/10.1016/j.jpdc.2012.02.018 Metaheuristics on GPUs.

[89] Hartmut Pohlheim. 2011. GEATbx - Genetic and Evolutionary Algorithm Toolbox for use with Matlab. http://www.geatbx.com/.

[90] Juan Porta, Jorge Parapar, Ramón Doallo, Francisco F. Rivera, Inés Santé, and Rafael Crecente. 2013. High performance genetic algorithm for land use planning. *Computers, Environment and Urban Systems* 37 (2013), 45 – 58. https://doi.org/10.1016/j.compenvurbsys.2012.05.003

[91] Tomas Potuzak. 2014. Distributed/Parallel Genetic Algorithm for Road Traffic Network Division for Distributed Traffic Simulation. In *Intelligent Distributed Computing VII*, Filip Zavoral, Jason J. Jung, and Costin Badica (Eds.). Springer International Publishing, Cham, 151–156.

[92] T. Potuzak. 2014. Parallelization Possibilities of a Genetic Algorithm for Road Traffic Network Division for Distributed/Parallel Environment. In *2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*. 211–218. https://doi.org/10.1109/DS-RT.2014.33

[93] T. Potuzak. 2015. Distributed/Parallel Genetic Algorithm for Road Traffic Network Division Using Step Parallelization. In *2015 4th Eastern European Regional Conference on the Engineering of Computer Based Systems*. 67–74. https://doi.org/10.1109/ECBS-EERC.2015.19

[94] P Prakash, KG Darshaun, Medidhi Venkata Ganesh, B Vasudha, et al. 2017. Fog Computing: Issues, Challenges and Future Directions. *International Journal of Electrical and Computer Engineering (IJECE)* 7, 6 (2017), 3669–3673.

[95] Dave Radford. 2016. *A Comparative Analysis of the Performance of Scalable Parallel Patterns Applied to Genetic Algorithms and Configured for NVIDIA GPUs*. Ph.D. Dissertation. The University of Guelph.

[96] Sergio Ramírez-Gallego, Alberto Fernández, Salvador García, Min Chen, and Francisco Herrera. 2018. Big Data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce. *Information Fusion* 42 (2018), 51–61. https://doi.org/10.1016/j.inffus.2017.10.001

[97] Gerhard Reinelt. 1991. TSPLIBâĂŤA Traveling Salesman Problem Library. *ORSA Journal on Computing* 3, 4 (1991), 376–384. https://doi.org/10.1287/ijoc.3.4.376 arXiv:https://doi.org/10.1287/ijoc.3.4.376

[98] J. Ren, Y. Pan, A. Goscinski, and R. A. Beyah. 2018. Edge computing for the internet of things. *IEEE Network* 32, 1 (Jan 2018), 6–7. https://doi.org/10.1109/MNET.2018.8270624

[99] V. Roberge, M. Tarbouchi, and G. Labonte. 2013. Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning. *IEEE Transactions on Industrial Informatics* 9, 1 (Feb 2013), 132–141. https://doi.org/10.1109/TII.2012.2198665

[100] V. Roberge, M. Tarbouchi, and F. Okou. 2014. Strategies to Accelerate Harmonic Minimization in Multilevel Inverters Using a Parallel Genetic Algorithm on Graphical Processing Unit. *IEEE Transactions on Power Electronics* 29, 10 (Oct 2014), 5087–5090. https://doi.org/10.1109/TPEL.2014.2311737

[101] Ozgur Koray Sahingoz. 2014. Generation of Bezier Curve-Based Flyable Trajectories for Multi-UAV Systems with Parallel Genetic Algorithm. *Journal of Intelligent & Robotic Systems* 74, 1 (01 Apr 2014), 499–511. https://doi.org/10.1007/s10846-013-9968-6

[102] Carolina Salto and Enrique Alba. 2015. Adapting Distributed Evolutionary Algorithms to Heterogeneous Hardware. *Trans. Computational Collective Intelligence* 19 (2015), 103–125. https://doi.org/10.1007/978-3-662-49017-4_7

[103] Pasquale Salza, Filomena Ferrucci, and Federica Sarro. 2016. Elephant56: Design and Implementation of a Parallel Genetic Algorithms Framework on Hadoop MapReduce. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion (GECCO '16 Companion)*. ACM, New York, NY, USA, 1315–1322. https://doi.org/10.1145/2908961.2931722

[104] John Schroeder. 2008. Remote optimization. In *Process Automation Services & Capabilities*. ABB, 13–16.

[105] Birgitt SchÃűnfisch and AndrÃĴ de Roos. 1999. Synchronous and asynchronous updating in cellular automata. *Biosystems* 51, 3 (1999), 123 – 143. https://doi.org/10.1016/S0303-2647(99)00025-8

[106] Vitaly Shaferman and Tal Shima. 2015. Tracking Multiple Ground Targets in Urban Environments Using Cooperating Unmanned Aerial Vehicles. *Journal of Dynamic Systems, Measurement, and Control* 5 (2015), 051010–051010–11. Issue 137. https://doi.org/10.1115/1.4028594

[107] A. Shayeghi, D. Gotz, J. B. A. Davis, R. Schafer, and R. L. Johnston. 2015. Pool-BCGA: a parallelised generation-free genetic algorithm for the ab initio global optimisation of nanoalloy clusters. *Phys. Chem. Chem. Phys.* 17 (2015), 2104–2112. Issue 3. https://doi.org/10.1039/C4CP04323E

[108] Z. Shen, K. Wang, and F. Y. Wang. 2013. GPU based Non-dominated Sorting Genetic Algorithm-II for multi-objective traffic light signaling optimization with agent based modeling. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 1840–1845. https://doi.org/10.1109/ITSC.2013.6728496

[109] Othman Soufan, Dimitrios Kleftogiannis, Panos Kalnis, and Vladimir B. Bajic. 2015. DWFS: A Wrapper Feature Selection Tool Based on a Parallel Genetic Algorithm. *PLOS ONE* 10, 2 (02 2015), 1–23. https://doi.org/10.1371/journal.pone.0117988

[110] Daniel H. Stolfi and Enrique Alba. 2014. Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms. *Applied Soft Computing* 24 (2014), 181 – 195. https://doi.org/10.1016/j.asoc.2014.07.014

[111] Daniel H. Stolfi and Enrique Alba. 2017. Computing New Optimized Routes for GPS Navigators Using Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '17)*. ACM, New York, NY, USA, 1240–1247. https://doi.org/10.1145/3071178.3071193

[112] Janko Straßburg, Christian Gonzàlez-Martel, and Vassil Alexandrov. 2012. Parallel genetic algorithms for stock market trading rules. *Procedia Computer Science* 9 (2012), 1306 – 1313. https://doi.org/10.1016/j.procs.2012.04.143 Proceedings of the International Conference on Computational Science, ICCS 2012.

[113] C. Sunitha and I. Jeevitha. 2015. A Review on Genetic Algorithm Practice in Hadoop MapReduce. *International Journal of Science Technology & Engineering* 2, 5 (12 2015), 150–155.

[114] Y. Takahashi, Y. Nojima, and H. Ishibuchi. 2015. Rotation effects of objective functions in parallel distributed multiobjective fuzzy genetics-based machine learning. In *2015 10th Asian Control Conference (ASCC)*. 1–6. https://doi.org/10.1109/ASCC.2015.7244890

[115] El-Ghazali Talbi. 2009. *Metaheuristics: From Design to Implementation*. Wiley.

[116] El-Ghazali Talbi. 2015. Parallel Evolutionary Combinatorial Optimization. In *Handbook of Computational Intelligence*.

[117] El-Ghazali Talbi and Geir Hasle. 2013. Metaheuristics on GPUs. *J. Parallel Distrib. Comput.* 73, 1 (2013), 1–3. https://doi.org/10.1016/j.jpdc.2012.09.014

[118] Diya Thomas and Binsu C. Kovoor. 2018. A Genetic Algorithm Approach to Autonomous Smart Vehicle Parking system. *Procedia Computer Science* 125 (2018), 68 – 76. https://doi.org/10.1016/j.procs.2017.12.011 The 6th International Conference on Smart Computing and Communications.

[119] Marco Tomassini. 1999. Parallel and Distributed Evolutionary Algorithms: A Review.

[120] Umut Tosun, Tansel Dokeroglu, and Ahmet Cosar. 2013. A robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem. *International Journal of Production Research* 51, 14 (2013), 4117–4133. https://doi.org/10.1080/00207543.2012.746798 arXiv:https://doi.org/10.1080/00207543.2012.746798

[121] Jamal Toutouh and Enrique Alba. 2017. Parallel multi-objective metaheuristics for smart communications in vehicular networks. *Soft Computing* 21, 8 (01 Apr 2017), 1949–1961. https://doi.org/10.1007/s00500-015-1891-2

[122] A. J. Umbarkar and M. S. Joshi. 2013. REVIEW OF PARALLEL GENETIC ALGORITHM BASED ON COMPUTING PARADIGM AND DIVERSITY IN SEARCH SPACE. *ICTACT Journal on Soft Computing* 3, 4 (2013), 615–622. https://www.ingentaconnect.com/content/doaj/09766561/2013/00000003/00000004/art00007

[123] José Valente de Oliveira, Sérgio Baltazar, and Helder Daniel. 2017. On asynchronous parallelization of order-based GA over grid-enabled heterogenous commodity hardware. *Soft Computing* 21, 21 (01 Nov 2017), 6351–6368. https://doi.org/10.1007/s00500-016-2190-2

[124] Abhishek Verma, Xavier Llorà, David E Goldberg, and Roy H Campbell. 2009. Scaling genetic algorithms using mapreduce. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference On*. IEEE, 13–18.

[125] Pablo Vidal and Enrique Alba. 2010. *Cellular Genetic Algorithm on Graphic Processing Units*. Springer Berlin Heidelberg, Berlin, Heidelberg, 223–232. https://doi.org/10.1007/978-3-642-12538-6_19

[126] Pablo Vidal and Enrique Alba. 2010. A multi-GPU implementation of a Cellular Genetic Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010*. 1–7. https://doi.org/10.1109/CEC.2010.5586530

[127] Darrell Whitley and Timothy Starkweather. 1990. GENITOR II: a distributed genetic algorithm. *Journal of Experimental & Theoretical Artificial Intelligence* 2, 3 (1990), 189–214. https://doi.org/10.1080/09528139008953723 arXiv:https://doi.org/10.1080/09528139008953723

[128] Jonathan Wright and Ali Alajmi. 2005. The robustness of genetic algorithms in solving unconstrained building optimisation problems. In *Proceedings of the 7th IBPSA Conference: Building Simulation*. 15–18.

[129] Rakesh Yadav. 2015. *Genetic Algorithms using Hadoop MapReduce*. Ph.D. Dissertation. National Institute of Technology Rourkela.

[130] C Yang, B Chen, and S Zhang. 2011. Design and implementation of general integrated optimization design software SiPESC.OPT. 20 (01 2011), 42–48.

[131] Chunfeng Yang, Haijiang Li, Yacine Rezgui, Ioan Petri, Baris Yuce, Biaosong Chen, and Bejay Jayan. 2014. High throughput computing based distributed genetic algorithm for building energy consumption optimization. *Energy and Buildings* 76 (2014), 92 – 101. https://doi.org/10.1016/j.enbuild.2014.02.053

[132] Zhaosheng Yang, Duo Mei, Qingfang Yang, Huxing Zhou, and Xiaowen Li. 2014. Traffic Flow Prediction Model for Large-Scale Road Network Based on Cloud Computing. *Mathematical Problems in Engineering* 2014 (2014), 8 pages.

[133] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report.

Received July 2018