processes

MDPI

Review

A Systematic Review of Multi-Objective Evolutionary Algorithms Optimization Frameworks

Andrei Pătrăușanu *, Adrian Florea, Mihai Neghină, Alina Dicoiu and Radu Chiș

Department of Computer Science and Electrical Engineering, Lucian Blaga University of Sibiu, 4 Emil Cioran, Str., 550025 Sibiu, Romania; adrian.florea@ulbsibiu.ro (A.F.); mihai.neghina@ulbsibiu.ro (M.N.); alina.dicoiu@ulbsibiu.ro (A.D.); radu.chis@ulbsibiu.ro (R.C.)

* Correspondence: andrei.patrausanu@ulbsibiu.ro

Abstract: The study of evolutionary algorithms (EAs) has witnessed an impressive increase during the last decades. The need to explore this area is determined by the growing request for design and the optimization of more and more engineering problems in society, such as highway construction processes, food and agri-technologies processes, resource allocation problems, logistics and transportation systems, microarchitectures, suspension systems optimal design, etc. All of these matters refer to specific highly computational problems with a huge design space, hence the obvious need for evolutionary algorithms and frameworks, or platforms that allow for the implementing and testing of such algorithms and methods. This paper aims to comparatively analyze the existing software platforms and state-of-the-art multi-objective optimization algorithms and make a review of what features exist and what features might be included next as further developments in such tools, from a researcher's perspective. Additionally, it is essential for a framework to be easily extendable with new types of problems and optimization algorithms, metrics and quality indicators, genetic operators or specific solution representations and results analysis and comparison features. After presenting the most relevant existing features in these types of platforms, we suggest some future steps and the developments we have been working on.

Keywords: multi-objective optimization; evolutionary algorithms; frameworks; software; platforms

Citation: Pătrăușanu, A.; Florea, A.; Neghină, M.; Dicoiu, A.; Chiș, R. A Systematic Review of Multi-Objective Evolutionary Algorithms Optimization Frameworks. *Processes* **2024**, *12*, 869. https://doi.org/10.3390/pr12050869

Academic Editor: Chien-Chih Wang

Received: 5 April 2024
Revised: 22 April 2024
Accepted: 23 April 2024
Published: 26 April 2024

1. Introduction

1.1. Evolutionary Computing

Over the last decades, the contributions to the field of multi-objective evolutionary algorithms (MOEAs) have grown exponentially. The growing need for real time automatization and optimization has become an imperative request in the current engineering area. For instance, the development of microarchitectures is a very relevant example when it comes down to searching for optimal solutions in the set of all possible design solutions. If such an architecture were to have 15 parameters, each of them taking 5 possible values, the total number of combinations would be higher than 30.5×10^9. An exhaustive search is completely inefficient, since the search space of 30.5 billion combinations is gigantic, especially the time required to simulate each combination. Similarly, the approach of hardware implementation and testing would not work, due to the huge amount of costs. Evolutionary computing and its specific algorithms can represent solutions to combinatorial problems characterized by huge complexities, especially those where the time for thorough problem analysis decreases. Evolutionary computing began by borrowing and applying ideas from the biological evolutionary theory, like natural selection and genetic inheritance, particle swarms-based and ant colony-based algorithms, red deer mating customs, gray wolf attacks, etc., into computer science and engineering and continues to look toward new biological research findings for inspiration. Furthermore, in most cases, a problem needs to be optimized considering at least two objectives. These objectives usually have an

antagonist behavior. The improvement of one of these objectives might lead to a decrease in the performance of the other objectives, which makes the optimization process more difficult to achieve.

There are four major categories of MOEAs which differ depending on the problem representation model:

- Evolutionary strategies: Proposed and developed starting in 1970 by Ingo Reichenberg and Hans-Paul Schwefel [1]. The representation of individuals consists of floating-point numbers and the main characteristic is the variable step of mutation.
- Genetic programming: Developed from 1990 by John Koza [2]. Data is represented as trees with crossover being accomplished through prune and graft operations. It fits very well in languages such as LISP.
- Evolutionary programming: Introduced in 1960 by J.L Fogel [3]. It is very similar to genetic programming but the representation is performed by the use of a finite state machine and is mutation based (no crossover).
- Genetic algorithms: These algorithms are the most widely known type of evolutionary algorithms. The mechanism for formalizing the prediction of individuals' quality in successive generations, the so-called Schemata Theory was introduced, with their help, by John Holland in 1970 [4].

Given the wide interest of researchers in the current scientific community [5,6] for genetic algorithms it is really necessary to have one or more software tools that provide the chance to analyze and compare these algorithms. Our goal is to increase awareness about the importance of developing such tools by emphasizing the needs that the researchers' community might find relevant to be included by the framework software developers. Moreover, it is essential to have a common platform for benchmarking existing algorithms. Also, such a platform might offer some very important details about the actual implementations of the algorithms, since their code is very rarely made public by the authors. Therefore, we present a systematic review of the current state-of-the-art frameworks, analyzing their features from multiple perspectives such as algorithms and problems library, quality indicators, and possibilities of parallelization, but also their limitations and niches to be exploited in future research and development, etc.

All of these algorithms and developments are perfectly applicable to smart industrial engineering processes. Industry 5.0 integrates more and more optimization and automatization methods in different fields, such as quality control, energy management, human–robots collaboration, food and agri-technology processes, physical or human resource allocation problems in cloud computing, construction or manufacturing industries, logistics and transportation systems, the design of microarchitectures and 3D micro-architected implants, etc.

The rest of this paper is structured as follows: Section 2 presents the research strategy and how this work might prove to be useful for all researchers who are studying this area; Section 3 briefly describes some state-of-the art evolutionary algorithms; the most well-known multi-objective optimization frameworks are presented in Section 4. This is divided in five categories, each of them corresponding to the description of one such framework; the last section is a short summary of what these tools currently provide, the limitations and constraints they have and future developments (e.g., new MOEAs) we intend to integrate in FADSE 2.0 (Framework for Automatic Design Space Exploration) software tool [7].

1.2. Objectives

We have two goals: (1) to identify the most used and cited frameworks in the scientific literature that are used to develop and test evolutionary algorithms and, (2) to perform a review of these frameworks and multi-objective optimization evolutionary algorithms regarding benefits, constraints, and limitations. Based on our experience with these tools, we built the analysis around the following research questions:

RQI: Which are the most useful tools/platforms available for working with Multi-Objective Optimization Evolutionary Algorithms and their target problems?
RQII: What essential features do they provide?
RQIII: Which are the constraints and limitations?
RQIV: How easily to extensible and maintainable are these tools from a software perspective?

2. Research Strategy

To address the issues presented above, we searched and analyzed several MOEA frameworks from the Scopus database, based on their popularity (number of references) in the scientific community. In Table 1, we present the 10 most cited works ([8–17]) from the Scopus database related to multi-objective optimization algorithms and platforms.

Table 1. Top 10 most cited papers from Scopus database related to multi-objective optimization algorithms and platforms.

Authors	Title	Source	Year	Times Cited
Tian, Y., Cheng, R., Zhang, X., Jin, Y. [8]	PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [Educational Forum]	IEEE Computational Intelligence Magazine, 12(4), pp. 73–87, 8065138	2017	1406
Evins, R. [9]	A review of computational optimization methods applied to sustainable building design	Renewable and Sustainable Energy Reviews, 22, pp. 230–245	2013	551
Altiparmak, F., Gen, M., Lin, L., Paksoy, T. [10]	A genetic algorithm approach for multi-objective optimization of supply chain networks	Computers and Industrial Engineering, 51(1), pp. 196–215	2006	496
Zhu, Z., Zhang, G., Li, M., Liu, X. [11]	Evolutionary Multi-Objective Workflow Scheduling in Cloud	IEEE Transactions on Parallel and Distributed Systems, 27(5), pp. 1344–1357, 7127017	2016	324
Kaur, K., Garg, S., Aujla, G.S., Kumar, N., Rodrigues, J.J.P.C., Guizani, M. [12]	Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay	IEEE Communications Magazine, 56(2), pp. 44–51	2018	301
Alba, E., Luque, G., Nesmachnow, S. [13]	Parallel metaheuristics: Recent advances and new trends	International Transactions in Operational Research, 20(1), pp. 1–48	2013	238
Liu, S., Wang, S., Zhu, F., Zhang, J., Krishnan, R. [14]	HYDRA: Large-scale social identity linkage via heterogeneous behavior modeling	Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 51–62	2014	229
Wang, Y., Liu, H., Zheng, W., Xia, Y., Li, Y., Cheng, P., Guo, K., Xie, H. [15]	Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning	IEEE Access, 7, pp. 39974–39982, 8676306	2019	209

Table 1. *Cont.*

Authors	Title	Source	Year	Times Cited
Liu, Q., Cai, W., Shen, J., Fu, Z., Liu, X., Linge, N. [16]	A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment	Security and Communication Networks, 9(17), pp. 4002–4012	2016	199
Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., Hu, S. [17]	Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT	Future Generation Computer Systems, 93, pp. 278–289	2019	196

Apart from those already mentioned, there are many more frameworks proposed in the literature, each of them trying to overcome the weaknesses of its predecessors. In the work presented in [18], the authors proposed Java-based open-source Design Space Explorer, which includes a series of state-of-the art algorithms. In the original paper an automotive domain case-study is shown in order to demonstrate the applicability of the tool. The FADSE (Framework for Automatic Design Space Exploration) software tool, developed by the researchers in [7] is focused on automatic design space exploration for multicore and manycore systems, taking into account the continuing growth of computing systems complexity. Developed in Java, the framework is flexible and provides easy extension and portability. Most of the applications developed in our research group were mapped on FADSE and we are currently in the process of extending it with new algorithms, problems, or quality indicators. In the work presented in [19], a C++ template library, which provides multi-objective optimization, is presented. Due to its object-oriented design, it enhances easy extensibility for developing different hybrid optimization methods. Moreover, the integration of new real-world problems is facilitated by the interfaces and template design. The work presented in [20] introduced the idea of a so-called AutoMOEA algorithm. The goal is to achieve and develop new algorithms that might outperform the existing MOEAs in terms of performance and quality on continuous and combinatorial optimization problems.

The analyzed frameworks are not limited to the engineering and computer science field. Multi-objective optimization platforms are proposed in the literature for many different other domains, such as molecular drug-design evolution—AIDD [21]—or metabolism absorption and toxicology analysis—ChemMORT [22]. In Section 5.2, we broaden the discussion to present applications in additional fields such as health, bioinformatics, or finance.

Some other notable such tools are implemented in the following programming languages: Java—[23,24]; C/C++—[25,26]; C#—[27].

Bibliometric Study

In order to achieve a thorough analysis of the presented aim, we also conducted a bibliometric study of the papers published in this area. Figure 1 summarizes the main steps we employed as follows:

1. Choosing the research criteria: We decided to select research papers from the Scopus database referring to "multi-objective", "algorithms", "optimization" and "platforms".
2. Choosing the time period: We selected the "2005–present" period to provide our study a higher relevance. Prior to 2005, there was a significantly smaller number of publications than after 2005.
3. Analysis of results: An investigation of over 1760 papers returned by the criteria from steps 1 and 2. This analysis included the number of citations, the interest, and the relevance of the paper subject and the most frequent of the selected keywords.

4. Pre-processing data: Before using the data gathered from Scopus, we had to preprocess some of the keywords and syntaxes. In most cases, we aggregated two or more terms referring to the same concept, but still kept the total number of occurrences of all of them. For example, we combined "genetic algorithm" and "genetic-algorithms" as "genetic algorithm". Similarly, instead of "energy consumption", "energy distribution", "energy optimization" we used "energy utilization".
5. Analysis of occurrences: An investigation of the key terms to have strong correlations with each other, as indicated by authors from the Scopus database. We used the VOSviewer tool [28] in order to perform these advanced clustering correlations. We selected only the keywords that occurred at least 35 times. Moreover, by using the same tool, we generated in Figure 2 a map of the current research trends in scientific literature. The 4 clusters obtained are presented in Tables 2–5.

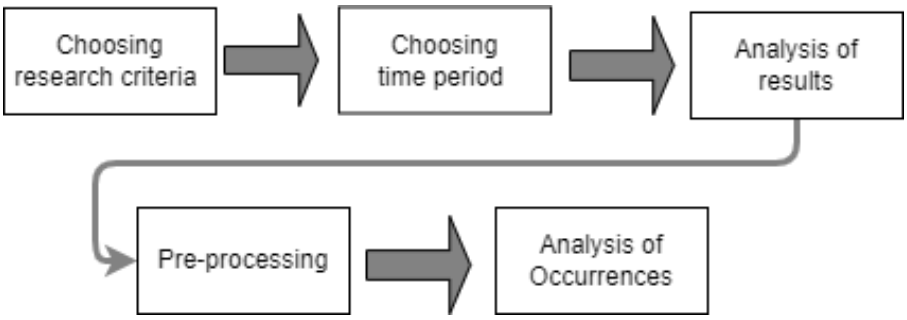

Figure 1. Steps of the bibliometric study.

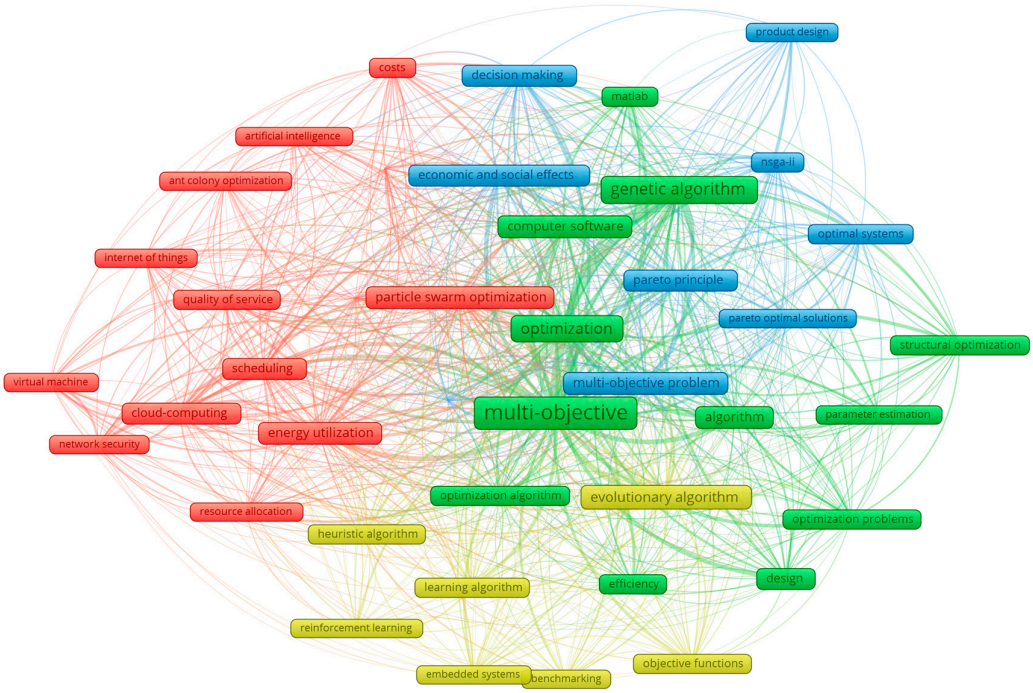

Figure 2. Map of current research trends.

The first cluster (red) focuses on swarm-based algorithms applied in different optimization areas like resource allocation, cloud computing, the Internet of Things, and efficient energy utilization. The keywords belonging to the second cluster (green) gravitate around multi-objective genetic algorithms applied in design space exploration and optimization problems from the industrial engineering domain, using MATLAB as an evaluation platform. The third cluster (blue) contains the keywords specific to Pareto dominance for multi-objective optimization problems with NSGA-II [29] as the main algorithm used in

decision-making or for product design with specific constraint problems. Finally, the fourth cluster (yellow) focuses on evolutionary algorithms and other heuristics or learning algorithms for problem solving and optimization, using benchmarking on standard platforms, but this could also be applied in engineering education by simulation and virtual reality environments required by digital twin transition (green and digital). Tables 2–5 shows the clustering of frequent keywords in that 1769 articles found.

Table 2. VOSviewer cluster 1.

Cluster 1—Swarm-Based Algorithms		
ID	**Keyword**	**Occurrences**
1	ant colony optimization	36
2	artificial-intelligence	49
3	cloud-computing	116
4	costs	60
5	energy utilization	173
6	integer programming	39
7	internet of things	39
8	network security	42
9	particle swarm optimization	168
10	performance	44
11	quality of service	61
12	resource allocation	40
13	scheduling	131
14	virtual machine	37

Table 3. VOSviewer cluster 2.

Cluster 2—Multi-Objective Genetic Algorithms		
ID	**Keyword**	**Occurrences**
1	algorithm	152
2	computer software	195
3	design	105
4	efficiency	53
5	genetic algorithm	568
6	matlab	84
7	multi-objective	1273
8	optimization	459
9	optimization algorithm	91
10	optimization problems	86
11	parameters estimation	50
12	structural optimization	77

Table 4. VOSviewer cluster 3.

Cluster 3—Pareto Dominance for MO Problems		
ID	**Keyword**	**Occurrences**
1	constrained optimization	44
2	decision making	99
3	economic and social effects	101
4	multi-objective problem	173
5	nsga-ii	70
6	optimal systems	62
7	pareto optimal solutions	46
8	pareto principle	116
9	product design	50

Table 5. VOSviewer cluster 4.

Cluster 4—Categories of Algorithms		
ID	**Keyword**	**Occurrences**
1	benchmarking	38
2	embedded systems	45
3	evolutionary algorithm	231
4	heuristic algorithm	53
5	learning algorithm	77
6	objective functions	55
7	reinforcement learning	35

Apart from the main focus of each cluster, from Tables 2–5 we can also extract different categories of problems which the provided algorithms can be applied on. These problems, suggested by the VOSviewer analysis, are shown in Table 6. Certainly, the keywords in Table 6 represent the general terms for such optimization areas. They can be divided into many other sub-categories.

Table 6. Generated problems from the VOSviewer analysis.

Cluster 4—Evolutionary Algorithms		
ID	**Problem**	**Occurrences**
1	costs optimization	60
2	energy utilization	173
3	internet of things	39
4	network security	42
5	resource allocation	40
6	scheduling	131
7	economic and social effects	101
8	embedded systems	45

3. State-of-the-Art Algorithms

Among the evolutionary algorithms discussed in the literature, two of the most successful are NSGA-II [29] and SPEA2 [30]. These two are part of the genetic algorithms category, and, by using similar concepts, they both offer an output of a population of non-dominated solutions. Despite this common non-dominated sorting and ranking approach, it is interesting to notice that both algorithms provide two different mechanisms for diversity preservation. After the current population is sorted in fronts of non-dominated solutions, if the number of required solutions from one front is less than the total size of that front, a diversity criterion is applied to select only the required number of individuals. In NSGA-II, this mechanism is called crowding distance, whilst in SPEA2 it is called environmental selection.

Over the time, a couple of improvements based on the idea of NSGA-II were proposed. For example, in CNSGA-II [31], the authors proposed a geometric distribution when selecting individuals from the non-dominated sorted population. This approach is intended to help diversity. NSGA-III [32,33] comes with a different approach, that is to use a reference-point based sorting approach. FD-NSGA-II [34] is a relatively new idea that incorporates the fuzzy logic in NSGA-II and SPEA2 in order to obtain a fuzzy Pareto dominance (FD) algorithm.

SMPSO [35] is another bio-inspired algorithm that aims to use the behavior of different particles in nature. It consists mainly of two parts: a cognitive component—a particle trying to improve its position in the swarm—relative to its own position; a social component—a particle trying to improve its own position—relative to the swarm leader position.

There are also some new attractive and fertile algorithms recently published in the literature, such as RDA—Red Deer Algorithm [36]—Gray-Wolf [37], or Hyena [38] which

demonstrated, according to the authors' experiments, the capability of yielding competitive results. These algorithms are also inspired by the reproduction of different species of animals in nature.

MOGBO [39] introduces a different approach compared to the existing algorithms, which is a gradient-based optimization procedure. The work presented in [40] proposes a memetic algorithm, APMA, where the crossover operator is usually replaced by a local search towards the last generations.

In the work presented in [41], the authors introduced the idea of solving optimization problems with irregular Pareto fronts. They designed two metrics based on parallel distance. These two metrics are then used in the mating and environmental selection phases to improve diversity and quality. It also helps in reducing the selection pressure. The difficulty of selection increases as the number of objectives increases as well. In the work presented in [42] a new algorithm called MaOEA-LAMG (Learning Assessment and Mapping Guidance), which uses past information, is presented. The motivation was to reduce the pressure selection if the number of objectives becomes bigger and bigger.

Another important aspect of such algorithms is the computing complexity. Ideally, if the problems to be optimized are complex, there should be some parallelization or task-management approach in order to speed-up the simulations. In the work presented in [43], a new interesting approach suggests dividing the original problem into multiple sub-tasks that can be executed separately. This would definitely help the main simulation, but it would be very demanding from the resources point of view.

Our research group introduced the CAFZGA—Controlled Apparent Front Zones Genetic Algorithm [44]—to exploit the vulnerabilities of MOEA, such as computational costs caused by Pareto-dominance. Evolutionary algorithms built on Pareto-dominance suffer from a loss of selection pressure as the number of objectives increases and the probability of finding non-dominated solutions in the population decreases. CAFZGA introduces a new genetic algorithm for multi-objective optimization based on Apparent Front Ranking and crowding distance. Computationally, CAFZGA is more efficient than GAs using Pareto-dominance because the set of support vectors for generating the Apparent Front Boundary is significantly smaller than the population. CAFZGA would enable a faster generation of results and the reduction of complexity (comparing with NSGA-II or other Pareto techniques) yielding a performance increase of ~250%, especially in the cases of problems with more than two objectives or in the situation where fitness evaluation would consume a lot of time (as in the case of processor architectures simulated on standardized benchmark suites with billions of instructions). Moreover, the Pareto-front does not need to be known beforehand, thus simplifying the evaluations of problems where a true Pareto-front is required for quality indicators.

In the next section, where we present frameworks, we also analyze if they provide the possibility to use algorithms such as the ones presented in this section.

4. Frameworks

Taking into consideration the number of citations per each tool and our continuous experiences with them, we selected 5 frameworks for analysis: PlatEMO [8], PyMOO [45], jMetal [46,47], Evolver [48], and FADSE [7].

It is worth mentioning that data from Table 7 were collected as of March 2024. Evolver was published in December 2023, hence the lack of citations.

Table 7. Analyzed frameworks.

Framework	No. of Scopus Citations	First Publication Date	Current Version	Language
jMetal	969	2006	6.1	Java
FADSE	43	2010	2.0	Java
PlatEMO	1372	2017	4.5	MATLAB
PyMOO	748	2020	0.6.0	Python
Evolver	0	2023	1.0.1	Java

Since the authors of jMetal have published several papers for each major update since 2011, such as the research presented in [49], they should be also taken into consideration when counting the number of citations. There are other two interesting libraries published by jMetal authors: jMetal.NET (2011, C#), whose development has been stopped and jMetalPy (2017, Python), whose development is still on-going.

Regarding FADSE, the authors did not specify any versioning method, so we assume that the current version is 2.0: version 1.0 corresponding to the work presented in [7,50] and version 2.0 corresponding to the work presented in [51].

Despite the flexibility provided by programming languages, it is also important to take into consideration the energy consumption, since evolutionary algorithms are usually run on complex computational problems. In the work presented in [52], the author shows the CO_2 consumption of several programming languages over the course of a high-demanding simulation in astrophysics. Based on this research, it is obvious that Python has a much higher CO_2 consumption level 10 times higher than Java and 100 times higher than C++.

In the work presented in [53], the authors presented comprehensive research over the energy consumption of several programming languages. One of their research questions was 'Is faster, greener?'. According to the energy equation Energy (J) = Power (W) $\times$ Time (s), a single general accepted answer cannot be given, since Power is not constant. Nevertheless, a reduction in execution time would imply a reduction in Energy consumption. In the same study, the authors present a Pareto set of optimal programming languages depending on different minimization objectives such as memory and time, energy and time, etc. Their conclusion is that compiled languages tend to be the fastest and most energy efficient ones. In the work presented in [54], a benchmarked classification of programming languages according to energy consumption is performed. The results show that C, C++, Rust, Ada and Java are the top 5 from this point of view, having the lowest energy consumption. It is also commonly known fact that C, C++ and Rust have been optimized and designed in this way. Similar findings were reported in an earlier study [55] that conducted an experiment on 14 compiled and interpreted programming languages by running tasks from the Rosetta Code programming chrestomathy. Their findings show that, among the compiled programming languages, C/C++ and Java offer the best performance, energy-wise, while JavaScript exhibits the highest energy savings among the interpreted languages, with Swift and Python being the most inefficient. The work presented in [56] performed a comparison of four different sorting algorithms, implemented in three programming languages, and concluded that both the choice of language and algorithm play a significant role in energy efficiency. They also emphasize that a large part of energy consumption is determined by the computational complexity of the solution. A comparison between the carbon footprints of four different nature-inspired optimization algorithms (Genetic Algorithm, Particle Swarm Optimization, Differential Evolution, Artificial Bee Colony) implemented in MATLAB is presented in the work presented in [57]. Their findings show Differential Evolution to be the greenest of the four, while Artificial Bee Colony has the highest energy consumption. The algorithm implementation, as well as the configuration parameters used when performing the experiment affect the overall environmental impact of each simulation. Thus, to mitigate the environmental concerns and enhance computational efficiency, optimization techniques can be applied at the algorithm level in order to reduce the computational complexity.

The presence of a graphical user interface (GUI) for frameworks represents an important feature which allows the user to interact and customize the values for each parameter in the application, to select and configure the problems and algorithms, perform the simulation, and inspect the results without having to write any new code. This improves the usability and increases the accessibility of the tool to non-IT specialists as well, as it eliminates the need for software development knowledge. Additionally, the GUI should display the evolution of solutions around the true Pareto-front and the quality indicators generation-by-generation over the entire run, which allows for the real-time assessment of the algorithm's behavior and the quality of solutions. The framework should provide

easy extensibility, meaning that the software architecture should be well structured, such that the interface level facilitates the easy integration of new algorithms and new problems that could enhance the tool. For example, in the work presented in [58] we designed a GUI framework for meta-optimization which can be easily used by any kind of user. In general, some of the essential software principles applied in UI design are: simplicity, usability, clarity and consistency.

Additionally, it is important to address the community support for each framework. This support might consist of the free use of platforms and will provide links to open-source code for community development, such as forums, FAQ or GitHub.

Moreover, despite the small or large number of optimization problems, a framework should also be reviewed from the type of problems perspective. The majority of the analyzed frameworks consists only of synthetic benchmark problems. Ideally, a framework must provide real-world scenario optimization problems, such as turning processes [59]. In the manufacturing industry, travelling salesman problems are extended by vehicle routing problems and the optimization of fuel consumption and CO_2 emissions [60], suspension design optimization [58], multi-objective optimizations of multicore processors architectures Sniper [61], M-SIM [62], etc.

4.1. PlatEMO

PlatEMO is developed in MATLAB, so it can be easily run by any specialist from Industrial Engineering with less programming skills on any operating system, provided that the MATLAB tool is installed. It consists of a GUI (Figure 3) which allows users to configure and run different simulations. The results are also displayed in a convenient visual way, as shown in Figure 3.

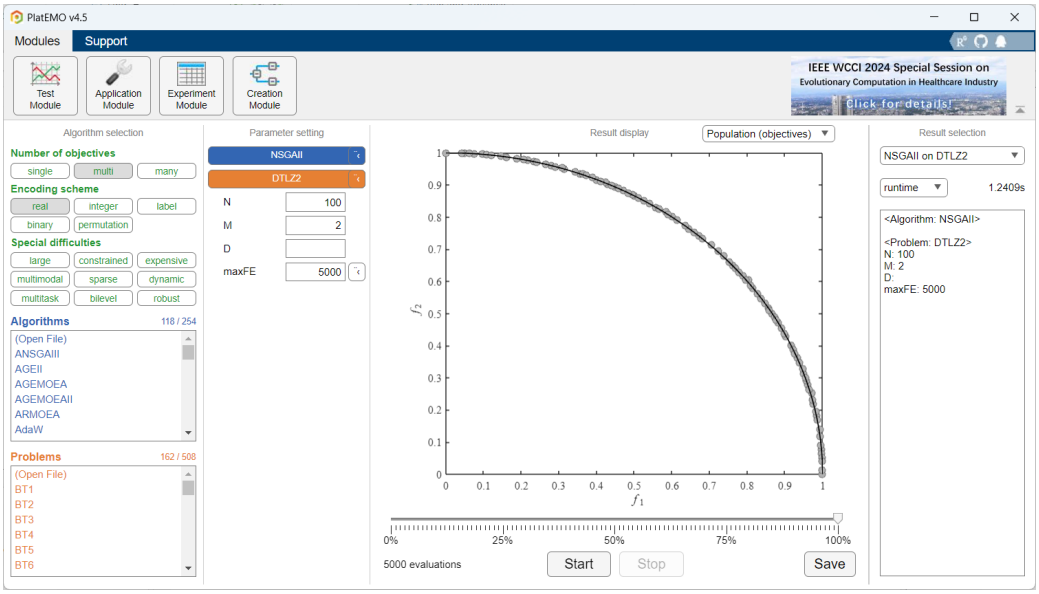

Figure 3. PlatEMO GUI.

The main features of PlatEMO are as follows.

- It is an open-source framework The source code is publicly available on GitHub for developers' contributions.
- It consists of a very rich and complex library of algorithms. There are more than 250 algorithms available in the application, divided in 4 categories: single objective, multi-objective (2 or 3 objectives) and many-objective (4 or more objectives). It consists of a large number of optimization problems—over 100 such problems.
- It provides easy extensibility. The software architecture is structured well so that the interface level allows an easy extensibility of the tool with new algorithms and

problems. In order to run different algorithms or problems, no more code is needed, since the GUI allows the user to select the desired options. This GUI allows the user to see the generation-by-generation evolution of solutions around the true Pareto-front and quality indicators of the results over the entire run.

- The software design allows the implementation of different meta-optimization algorithms, such as that presented in [63]. It would be possible to create at least two threads, each of them for one of the super-positioned algorithms, followed by starting the simulation on the same problem.

Features that are missing and what additions might be useful:

- A check-pointing mechanism to save individuals or populations periodically, in case some simulations fail before the end of the last generation.
- It does not address the mechanism of parallelization of the evaluation functions inside the individuals. This could generate very long runs in case of a complex and time-consuming problem.
- It does not include relatively new proposed algorithms such as RDA, Gray-Wolf, MOGBO, etc.
- It does not allow the user to change the range of the design variables, nor the constraints, where possible, for the selected optimization problems.

4.2. PyMOO

PyMOO (Python Multi-Objective Optimization) has been developed in Python, a language that has become very popular for research purposes due to its libraries and code level syntax, which provides a very handy way to work with mathematical intelligence or Artificial Intelligence concepts, such as: built-in functions, models selection, matrix operations, and machine learning algorithms and so on.

Its main features are as follows:

- It is an open-source framework. It is publicly available on GitHub for developers' contributions.
- It provides single-objective, multi-objective, and many-objective test problems.
- Optimization constraints are also available for the user.
- It contains gradient information provided by automatic differentiation [64].

The software architecture is divided in 3 main parts, each of them having separated sub-modules, which makes it easy for further extension with new algorithms and problems:

- Problems: single-, multi-, and many-objectives;
- Optimization (genetic operations): Crossover, Mutation, Survival, etc.;
- Analytics: Visualization, Performance Indicator, Decision Making. This extra feature, called Decision Making, provides some options to select one solution from the final Pareto-front, depending on different preferences.

PyMOO has strong parallelization of the solutions mechanism, such as: multi-threaded execution, distributed computing and vectorized computations. This might also help with integrating meta-optimization or superposition algorithms. As described earlier, for such an implementation, a multi-threaded architecture is required.

The authors benchmarked their implementations against some existing resources provided by the actual authors of state-of-the-art algorithms, as mentioned, so it can be considered an error-free tool. It provides a visual representation of the results: Pareto-front, solutions, and optimization problems characteristics.

Unfortunately, there are some weaknesses:

- In order to run new simulations, the user has to write new code.
- Every time a new algorithm or the same algorithm with different parameters needs to be started, a new object has to be created.
- This also applies to the other components: problems, quality indicators, etc. This is not necessarily a major problem, but some researchers might be interested in black-box usage of the tool: initial-config + results interpretation.

- The MOA library is extremely poor, containing less than 20 algorithms.
- The same happens with problems. There are few problems for testing those algorithms. Therefore, it does not include relatively new proposed algorithms such as: RDA, Gray-Wolf, MOGBO, etc.

Their results are indeed displayed in a GUI (Figure 4) as a Pareto-front, but it would have been better to display as much analysis as possible, like it is in PlatEMO (selectable quality indicators, generation-by-generation graphs).

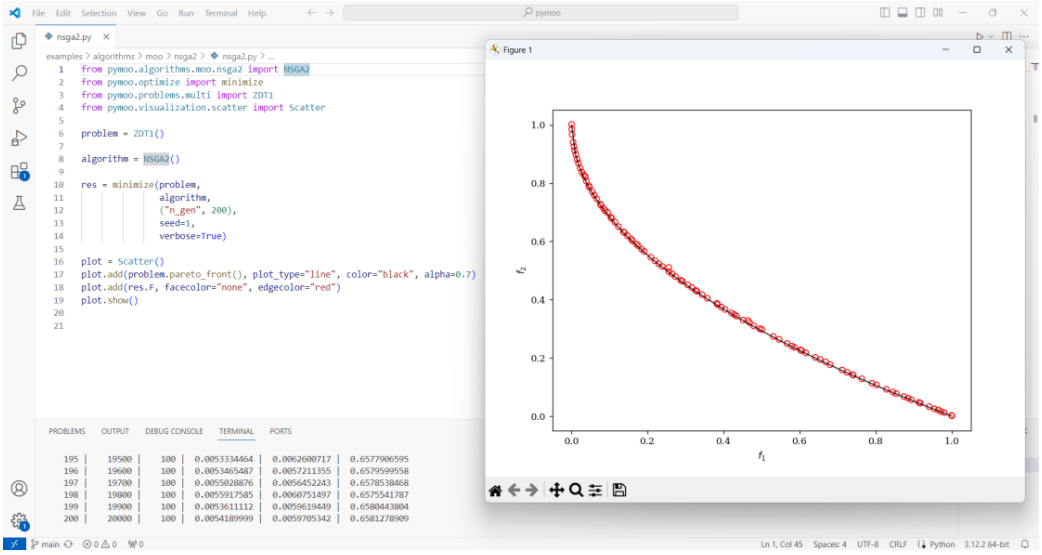

Figure 4. PyMOO application.

A checkpointing mechanism is missing and it might prove to be very useful.

4.3. jMetal

jMetal is a Java library, originally created in 2006, which have gone through major updates since its first publication. It is an open-source library rather than an application itself, which can be included in an existing application, or it be used as starting point for a project. It is publicly available on GitHub for developers' contributions.

It was also ported in Python, as a multi-objective library called JMetalPy [65].

Its main features are as follows:

- It contains a very rich set of algorithms and problems, around 100 state-of-the art algorithms.
- There are 3 possible categories: single-objective, multi-objective and many-objective algorithms.
- The software architecture consists of a very good object-oriented base design. Each category of algorithms inherits its own specific path, making it easy to maintain and easy to use. Therefore, a very easy extensibility with new algorithms and problems is possible.
- Documentation, code snippets, and examples of how to use different features and quality indicators are provided.
- It is relatively easy to integrate a meta-optimization and super-position mechanism within jMetal. The software design makes it feasible for such extensions.

The weaknesses shown by the platform are:

- There is no parallelization mechanism addressed.
- There is no checkpointing mechanism addressed.
- The results are not displayed in a visual manner. As mentioned earlier, it might be regarded as a library, but the end should be given a confirmation about the end of the simulation and the obtained results.

- It does not include relatively new proposed algorithms such as: Hyena, Gray-Wolf, MOGBO, etc.

4.4. Evolver

Evolver is a tool developed by a group of researchers that includes the authors of the original jMetal tool, as mentioned above. It implements the idea of meta-optimization: to find, using evolutionary algorithms, the optimum parameters configuration for an evolutionary algorithm that aims to optimize a problem. The full code is hosted on GitHub for contributions.

Since all the core of Evolver is in fact, the jMetal library, it comes with all the advantages and weaknesses from jMetal. Therefore, in the next paragraph we are only focusing on the extra features, such as meta-optimization, that Evolver provides.

Its main characteristics are:

A practical implementation idea of meta-optimization. It could very useful to know what the optimal configuration for a specific algorithm is before starting it. The algorithm itself might yield better or worse results, depending on the configuration. There are not many such tools in the literature that approach the meta-optimization area so such a concrete implementation is really useful.

- The algorithm, whose optimal configuration is to be found and is referred to as a Configurable Algorithm is given a training set. This set is actually a collection of optimization problems; hence the diversity of training is provided.
- It could be easily extended to work with any real-world engineering problem. As an example, in the original paper, the authors present a case-study of an injector design problem. As an objective of meta-optimization, a list of quality indicators is set to be minimized. This list is configurable.

The project consists of two parts:

- Evolver—the core of the meta-optimization approach (jMetal, problems, etc.);
- Evolver Dashboard—a user-friendly web application built with Python, which allows the user to configure and modify all of the parameters for meta-optimization. The web-interface provides a very detailed and relevant visualization of the results: generation-by-generation population, indicator values, time, evaluations, pareto-front, etc.
- The project comes also as a pre-built docker package to ease the interaction with the tool. It provides easy software extensibility.

The main weaknesses are as follows:

- The algorithmic idea of meta-optimization might take a lot of time to run, typically when the problem to optimize is very complex. An evolutionary algorithm itself might require a significant amount of time, let alone an algorithm which uses a population of such evolutionary algorithms.
- Considering the previous idea, checkpointing, individual reusability mechanisms, and evaluation parallelization should be addressed.
- A logging feature to save the progress or status of simulation at some points in time would be useful, as well. In the case of a long-run simulation, the evaluation of some individuals may fail or the simulation itself may fail, therefore, leading to unknown scenarios.

4.5. FADSE

FADSE (Framework for Automatic Design Space Exploration) is a Java framework which focuses on multi-objective optimization algorithms. As mentioned earlier, it was originally developed by Horia Calborean and further extended by Radu Chis within the ACAPS (Advanced Computer Architecture and Processing Systems—https://centers.ulbsibiu.ro/acaps/ (accessed on 12 March 2024) research group for his PhD thesis.

Its main features are as follows:

- An open-source framework that is based on jMetal library. Therefore, it comes with a rich library of algorithms and problems. It is publicly available on GitHub for developers' contributions.
- It includes single and multi-objective algorithms.
- Apart from the existing problems from jMetal, FADSE has been extended with some new engineering problems such as the multi-objective optimization of mono- and multi-core architectures as GAP—Grid ALU Processor [66]—SNIPER [61], or MSIM [62].
- It provides quality indicators and several monitoring reports to see the progress of the current simulations.
- It also provides a robustness mechanism including a SQL database connection in order to save the already simulated individuals. It is also tolerant to simulation failures due to the checkpointing mechanism.
- It is possible to use fuzzy logic rules in order to help algorithms to obtain a faster convergence. This fuzzy mechanism is easy to extend and manipulate by domain knowledge experts.
- There is a parallelization mechanism implemented in FADSE, such that, if connected to a computer network or high-performance computers, individuals are sent among this network and simulated at the same time.

There are also some things to be improved, such as:

- The main drawback of FADSE is that it uses jMetal version 0.8. This is one of the first versions of jMetal and it is outdated. The object-orientated design is very poor. Therefore, the extensibility of the tool is very difficult because of this poor software design. Since the jMetal library used is that old, FADSE misses many of the more recent algorithms.
- The same thing is valid for optimization problems and metrics.
- Moreover, all the changes have been made directly inside the code of this old meta-heuristics library over time. That means that an update to a recent version of jMetal is extremely difficult because of the lack of compatibilities between the current jMetal version included in FADSE and other recent jMetal versions.
- It does not provide a GUI, but the input to FADSE is given in a custom xml file. Nevertheless, FADSE provides features to run simulations on High-Performance Computers within Local Area Networks.
- Nevertheless, the documentation is missing, which makes it very difficult for developers to get started with FADSE.
- It is not possible to implement any meta-optimization or superposition mechanism because the software architecture was designed so that a simulation allows only one thread to be started for one optimization problem.

4.6. General Comparison

Table 8 synthesis the main features of previously analyzed frameworks.

In general, the comparison of the performance of some optimization algorithms is realized according to the degree of complexity of the algorithms (e.g., MN^3—NSGA, or MN^2—NSGA-II, SPEA2, where M is the number of objectives and N is the population size of genetic algorithm, MV—CAFZGA, where V is the size of support vectors), and is tested on synthetic benchmarks (ZDT1-6, SCH, KUR, DTLZ1–DTLZ9, WFG1-9, etc.) [8,30,31,44–48]. Actually, all analyzed frameworks use the synthetic benchmarks mentioned as test problems. The computational efficiency and potential environmental impact of using these algorithms depends on the language the platform has been written in but also on the complexity of the optimization problem (most of the time non-polynomial). On the other hand, regarding FADSE, the authors continuously extend the framework, which could run the synthetic benchmarks, but the focus is on real-life problems from the domain of computer architecture (automatic design space exploration of different single/multi-cores processors), industrial engineering (turning process, suspension optimization), energy sector (prediction of electricity consumption and production), etc. The authors have shown

that by using an automatic design space exploration process, the solutions found are far superior to a manual run. In the work presented in [50], the automatic DSE run on the GAP simulator found configurations with twice the performance for the same energy consumption compared to the manual runs, while in the work presented in [51], the authors have shown that automatic DSE on complex actuator models has found configurations with better operation characteristics (smoother curves) with a 5% decrease in size, which translates into a more efficient design.

Table 8. General comparison of the selected 5 frameworks.

Feature/Framework	PlatEMO	PyMOO	jMetal	Evolver	FADSE
Multi-objective focus	✓	✓	✓	✓	✓
Rich algorithms library	✓	✗	✓	✓	✓
Rich problems library	✓	✗	✓	✓	✓
Parallelization	✗	✗	✗	✗	✓
Checkpointing	✗	✗	✗	✗	✓
Easy extensibility	✓	✓	✓	✓	✗
GUI	✓	✗	✗	✓	✗
Open-source	✓	✓	✓	✓	✓
Comprehensive documentation	✓	✓	✓	✓	✗

In addition to presenting a comprehensive comparison of five of the most cited or promising multi-objective optimization frameworks, this paper aims to invite the research community into wider adoption and provides open contributions in developing such platforms, helping in researching and innovating any scientific domain. In Table 9 we present the number of contributors and the number of watchers for each platform. The advantage of such platforms, at least in the case of FADSE on which the authors are working, allows for the optimization of almost any type of problem, the only thing necessary being the development of connectors that map the solution of the problem in chromosomes specific to the genetic algorithms used.

Table 9. Community support for the selected 5 frameworks.

Framework	Link GitHub	# of Contributors	# of Watchers
PlatEMO	https://github.com/BIMK/PlatEMO (accessed on 12 March 2024)	12	45
PyMOO	https://github.com/anyoptimization/pymoo (accessed on 12 March 2024)	37	32
jMetal	https://github.com/jMetal/jMetal (accessed on 12 March 2024)	43	54
FADSE	https://github.com/horia-calborean/fadse (accessed on 12 March 2024)	3	6
Evolver	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00488 (accessed on 12 March 2024)	3	0

Although there are many quality indicators proposed in the literature [67,68], we decided to compare these frameworks by considering 10 commonly used indicators: Hypervolume (HV) [69], Two Sets Hypervolume Difference (TSHD) [70], Epsilon (EP) [68], Spread (SP) [29], Generalized Spread (GS) [71], NPS—Number of Pareto Solutions (NPS) [72], Mean Ideal Distance (MID) [73], Generational Distance (GD) [74], Inverted Generational Distance (IGD) [75], Coverage (C) [68]. Table 10 synthesizes the comparison among the framework according to the previously mentioned quality indicators.

Table 10. Quality indicator comparison of the selected 5 frameworks.

Quality Indicator	PlatEMO	PyMOO	jMetal	Evolver	FADSE
HV	✓	✓	✓	✓	✓
TSHD	✗	✗	✗	✗	✓
EP	✗	✗	✓	✓	✓
SP	✓	✗	✓	✓	✓
GS	✗	✗	✓	✓	✓
NPS	✗	✗	✗	✗	✗
MID	✗	✗	✗	✗	✗
GD	✓	✓	✓	✓	✓
IGD	✓	✓	✓	✓	✓
C	✓	✗	✓	✓	✓

5. Discussions

We do not aim to criticize at all the work of the authors we presented, but just to make a review from a researcher's perspective of all these tools, in order to extract general useful and handy features.. Therefore, the paragraphs in this section will not make any choice among these frameworks, but they will point out the essential features we consider as the most important ones.

5.1. Constraints and Limitations

Some common limitations we found are the lack of checkpointing, reusability mechanisms, and the parallelization of evaluations.

Checkpointing is very useful, especially in the case of long and complex optimization problems. If the process fails for some reason, it could be crucial to have a saved trace of the last successfully executed generation or individuals that could lead the researcher to the cause of failure. The electric power outage can be a problem as well for simulations that run for several hours or days.

Individual reusability can prove to be essential when the probability of generating similar individuals during a run is high. If a newly generated individual and its evaluated results are already stored in a database, it does not have to be evaluated again, therefore, gaining some time. If the retrieval percentage is too low, it will end up losing time because of the periodically pointless interrogations.

Parallelization is another key factor that should be considered. The evolutionary algorithms are embarrassingly parallel, because the individual could be run on different threads, cores, machines, etc., independently.

One typical constraint we identified, for example in PlatEMO, is the impossibility to change the range of the design variables or constraints before a simulation. They are hard-coded and should be changed inside the implementation. The same thing happens with algorithm parameters, like crossover, mutation, etc. By building the application in this closed manner, the user is constrained to run the algorithm and problems with mojstly the same parameters every time.

Another aspect worth considering is the interaction between the user (researcher) and the application. The usability of the frameworks differs between those equipped with a GUI and those lacking this feature. PlatEMO and Evolver provide a user-friendly interface, allowing researchers from various fields to fully take advantage of their capabilities, while requiring little-to-no programming expertise. This can support and accelerate research progress in a variety of domains by enabling researchers to focus on problem formulation and experimentation without the need for extensive programming assistance.

In contrast, running simulations and viewing results with the other frameworks that were discussed can only be achieved by writing new code. It is worth mentioning that frameworks like PyMOO and jMetal are designed as libraries that can be imported and used in software applications, rather than being standalone optimization tools. This is a constraint that limits their accessibility to users who are proficient in the particular

programming language in which the framework was developed, even if users just want to run some simulations, without further extending the library.

On the other hand, although FADSE does not yet have a GUI interface, it should be mentioned that it offers parallelism and the possibility of running on HPC systems and local distributed computer networks (LAN). One of the future developments aims to introduce the GUI in FADSE.

5.2. Other Specialized Optimization Existing Platforms

Although it was not our goal from the beginning, we consider it important for researchers to understand that advances in processing techniques have led to exponential growth in data, and especially in the field of biology and bioinformatics there is a big demanding in the development of large-scale bioinformatics experiments.

In the work presented in [76] the authors introduce Seq, an optimization framework for genomics freely available at https://seq-lang.org (accessed on 12 March 2024). Seq is a high-performance, domain-specific programming language for bioinformatics and computational genomics that bridges the ease of use and clarity of high-level languages such as Python or MATLAB with the performance of low-level languages such as C or C++. Seq has the syntax and semantics of the widely used high-level Python language, on top of which it adds genomics-specific but user-invisible language constructs, types, and optimizations. The creators of Seq claim that their framework is to bioinformatics what MATLAB is to many engineering problems—an accessible portal to the world of high-performance data science.

Another high-performance framework that uses scientific workflow technologies coupled with provenance data analytics for managing and analyzing bioinformatics experiments entitled BioWorkbench was presented in the work presented in [77]. Through a web interface it aims to support users in the process of specifying an experiment, determining the resources of a High-Performance Computing systems needed for the experiment, managing the data consumed and produced, and analyzing the results.

Although both source codes of the Seq and BioWorkbench are available on GitHub, the main difference between them and the 5 frameworks analyzed in this work, summarized in Table 7, is that both Seq and BioWorkbench are dedicated just to bioinformatics. FADSE and the other frameworks could be applied also to almost any type of problem, from different domains like: Aircraft Design, Routing in Communications Networks, Tracking Windshear, Game Playing, Medicine, Engineering application, Air Traffic Control, Automatic Design Space Exploration, Scheduling and logistics, Machine Learning, Expert system, VLSI Circuit Layout, Strike Force Allocation. Much closer to jMetal and FADSE is the MO-Phylogenetics framework [78], a software tool based on jMetalCPP and BIO++ with applications in bioinformatics, which is a phylogenetic inference software tool with multi-objective evolutionary metaheuristics, whose software developer is Antonio Nebro, who contributed to the jMetal framework.

A simple web search on the keywords “framework” and “health” generally provides articles on either the human performance optimization framework from a military or social perspective, or the simulation-based optimization frameworks to support logistics in the management of the healthcare sector and to avoid risks in critical response issues in emergencies. A novel general multi-level simulation-based optimization framework has been proposed in the work presented in [79], with the aim of supporting managerial decisions in the transition from standard forms of healthcare services to IoT-enabled systems targeting elderly care home assistance. The main idea is to use simulation to mimic system organization, rules, and behavior, whereas optimization is used to search for the allocation of both personnel and equipment to pursue optimality concerning resource availability. The framework is limited through the heuristics involved, and the authors mention local optimization techniques like Simulated Annealing. In the work presented in [80], some limitations are avoided by implementing Genetic Algorithms. However, the lack of multi-

objective optimization, the difficult implementation, and the lack of availability are some limitations of the framework.

In the work presented in [81], the authors used Genetic Algorithms and MATLAB software to develop a simulation-optimization framework that incorporates cash flow modelling and uncertainties in macroeconomic and microeconomic parameters in a supply chain planning problem while measuring the profitability. One of the limitations of this is that the authors do not apply multi-objective optimization in their study.

5.3. Future Developments

We presented some of the most complex tools available on the market by providing pros and cons, gathered from our working experience with them. As mentioned earlier, it is not possible to create a tool which satisfies all the needs for the research community. Nevertheless, we aim for the future, to build a tool that might include all the features we presented and we consider to be relevant for our needs. Our goal is to upgrade the existing FADSE [7], which was originally developed within our research group.

The main simulation flow of FADSE, including the data-base connection, problems wrappers, and the interface connection between modules is presented in Figure 5. Some of the ongoing or recently added developments are marked with red in the picture.

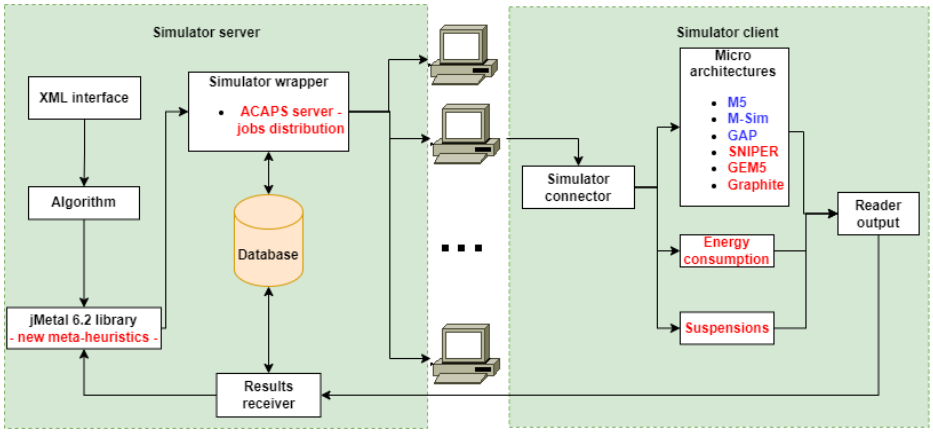

Figure 5. FADSE simulation flow.

The main extension of FADSE consists of replacing jMetal 0.8 with jMetal 6.2. Moreover, we are continuously extending jMetal with recently introduced algorithms such as: RDA, Gray-Wolf, Hyena, etc. More engineering problems will be added: design space exploration of multi- and many-core architectures or hardware accelerators, wine fermentation process optimization, energy production and consumption prediction, highway building process, suspension system optimization. A benchmarked implementation of CAFZGA [44], which uses the Apparent Front Ranking (AFR) approach instead of classic Pareto-based algorithms, will be included in FADSE. Additionally, the existing problems will be tested once more using the CAFZGA algorithm for more than two objectives. New variants of CAFZGA will be proposed, since the template function might change dynamically or it could be constructed on the fly, without prior knowledge of it. From the parallelization point of view, we aim to apply a load balancing mechanism to make FADSE distribute the simulations/individuals among the cores/machines across local area networks and high-performance computing in a very efficient way by introducing a job scheduling mechanism. One of the future developments aims to provide a graphical user interface into the FADSE framework to increase its accessibility degree. A feature to be introduced in FADSE will try to distinguish between optimization problems (with one or more objectives) according to their establishment before starting the optimization process and to suggest potential algorithms.

The main extended architecture of jMetal included in FADSE is presented in Figure 6, and we have already started to implement the mentioned features with red.

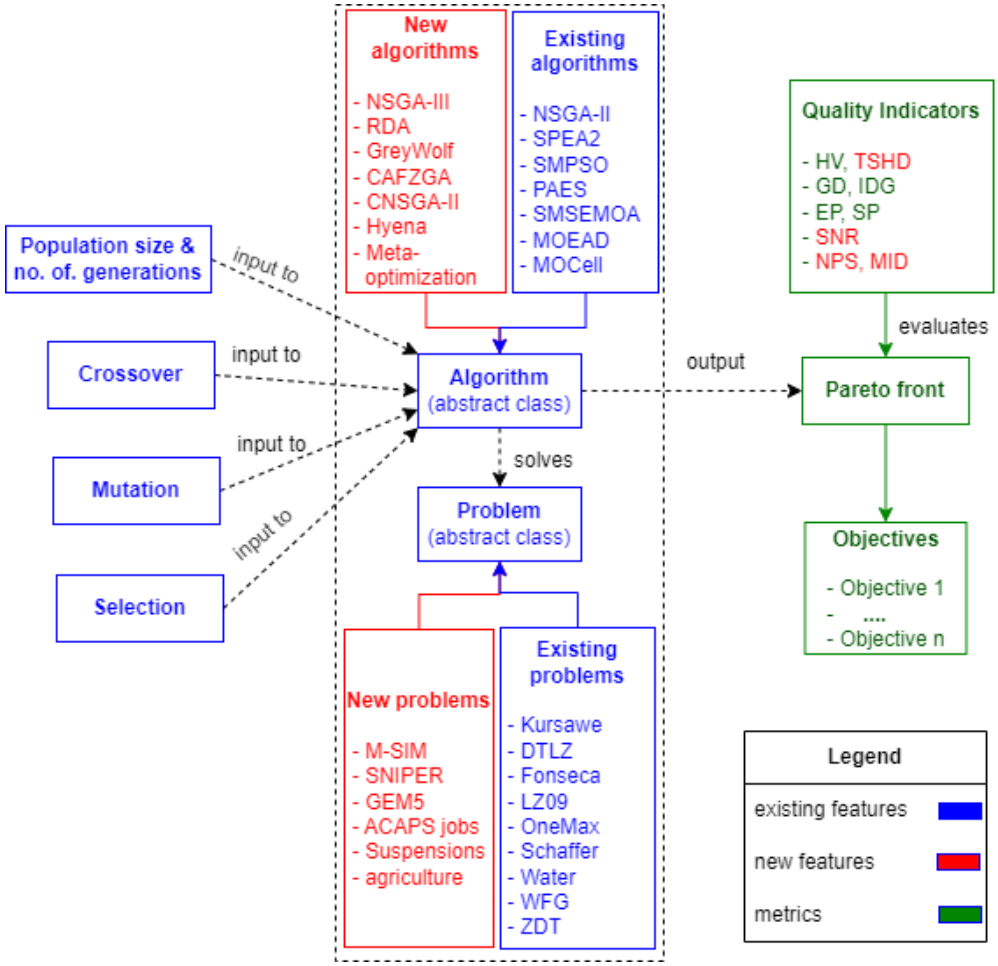

Figure 6. jMetal and FADSE extensions.

6. Concluding Remarks

A comprehensive comparison of multi-objective optimization frameworks is presented, covering 5 of the most cited or promising platforms. Obviously, none of the tools are accepted as perfect by the entire research community. Some researchers might find that one of the tools presents better for their needs than the others, or they might prefer a tool that has not even been mentioned in this review.

However, features such as flexibility, easy extensibility, parallelization, checkpointing, a good variety of metrics, efficient energy consumption and a reduced carbon footprint can be generally viewed as necessary and relevant in these kinds of platforms. Additionally, prioritizing usability by implementing a GUI that facilitates intuitive configuration, execution and result interpretation could also bring great benefits. Adherence to design principles such as clarity, consistency, and accessibility ensures an optimal user experience and increases efficiency in optimization research across different domains. The comparison of essential features allows for a good overview of the existing frameworks, as well as being a basis for evaluating the extent to which other existing (or future) tools contain these useful features. Although we highlighted important platforms available for working with Multi-Objective Optimization Evolutionary Algorithms, our analysis is not exhaustive in finding other important domains where such frameworks exist (bioinformatics, health, finance, etc.). From a software perspective, it is not always easy to extend and maintain these tools due to new software versions that appear, but also due to the new types of complex problems that need to be solved.

Funding: This work was partially developed in the project CoDEMO (Co-Creative Decision-Makers for 5.0 Organizations), grant number 101104819, an initiative supported by the Erasmus+ funding mechanism ERASMUS-EDU-2022-PI-ALL-INNO-EDU-ENTERP (Alliances for Education and Enterprises).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Beyer, H.-G.; Schwefel, H.-P. Evolution strategies—A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [CrossRef]
2. Koza, J. Non-Linear Genetic Algorithms for Solving Problems. U.S. Patent No. 4,935,877, 19 June 1990.
3. Fogel, L. *Intelligence through Simulated Evolution: Forty Years of Evolutionary Programming*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 1999.
4. Holland, J. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1975.
5. Sharma, S.; Chahar, V. A Comprehensive Review on Multi-objective Optimization Techniques: Past, Present and Future. *Arch. Comput. Methods Eng.* **2022**, *29*, 5605–5633. [CrossRef]
6. Liu, H.; Li, Y.; Duan, Z.; Chen, C. A review on multi-objective optimization framework in wind energy forecasting techniques and applications. *Energy Convers. Manag.* **2020**, *224*, 113324. [CrossRef]
7. Calborean, H.; Vintan, L. An automatic design space exploration framework for multicore architecture optimizations. In Proceedings of the 9th RoEduNet IEEE International Conference, Sibiu, Romania, 24–26 June 2010; pp. 202–207.
8. Tian, Y.; Cheng, R.; Zhang, X.; Jin, Y. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization. *IEEE Comput. Intell. Mag.* **2017**, *12*, 73–87. [CrossRef]
9. Evins, R. A review of computational optimisation methods applied to sustainable building design. *Renew. Sustain. Energy Rev.* **2013**, *22*, 230–245. [CrossRef]
10. Altiparmak, F.; Gen, M.; Lin, L.; Paksoy, T. A genetic algorithm approach for multi-objective optimization of supply chain networks. *Comput. Ind. Eng.* **2006**, *51*, 196–215. [CrossRef]
11. Zhu, Z.; Zhang, G.; Li, M.; Liu, X. Evolutionary Multi-Objective Workflow Scheduling in Cloud. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1344–1357. [CrossRef]
12. Kaur, K.; Garg, S.; Aujla, G.S.; Kumar, N.; Rodrigues, J.J.P.C.; Guizani, M. Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay. *IEEE Commun. Mag.* **2018**, *56*, 44–51. [CrossRef]
13. Alba, E.; Luque, G.; Nesmachnow, S. Parallel metaheuristics: Recent advances and new trends. *Int. Trans. Oper. Res.* **2013**, *20*, 1–48. [CrossRef]
14. Liu, S.; Wang, S.; Zhu, F.; Zhang, J.; Krishnan, R. HYDRA: Large-scale social identity linkage via heterogeneous behavior modeling. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Snowbird, UT, USA, 22–27 June 2014; pp. 51–62.
15. Wang, Y.; Liu, H.; Zheng, W.; Xia, Y.; Li, Y.; Cheng, P.; Guo, K.; Xie, H. Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access* **2019**, *7*, 39974–39982. [CrossRef]
16. Liu, Q.; Cai, W.; Shen, J.; Fu, Z.; Liu, X.; Linge, N. A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. *Secur. Commun. Netw.* **2016**, *9*, 4002–4012. [CrossRef]
17. Zhou, X.; Zhang, G.; Sun, J.; Zhou, J.; Wei, T.; Hu, S. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Gener. Comput. Syst.* **2019**, *93*, 278–289. [CrossRef]
18. Lukasiewycz, M.; Glaß, M.; Reimann, F.; Teich, J. Opt4J: A modular framework for meta-heuristic optimization. In Proceedings of the 13th Annual Conference Genetic and Evolutionary, Dublin, Ireland, 12–16 July 2011; pp. 1723–1730.
19. Shen, R.; Zheng, J.; Li, M. A hybrid development platform for evolutionary multi-objective optimization. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 1885–1892.
20. Bezerra, L.C.T.; Lopez-Ibanez, M.; Stutzle, T. Automatic Component-Wise Design of Multiobjective Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **2016**, *20*, 403–417. [CrossRef]
21. Jones, J.; Clark, R.D.; Lawless, M.S.; Miller, D.W.; Waldman, M. The AI-driven Drug Design (AIDD) platform: An interactive multi-parameter optimization system integrating molecular evolution with physiologically based pharmacokinetic simulations. *J. Comput. Mol. Des.* **2024**, *38*, 1–20. [CrossRef]
22. Yi, J.-C.; Yang, Z.-Y.; Zhao, W.-T.; Yang, Z.-J.; Zhang, X.-C.; Wu, C.-K.; Lu, A.-P.; Cao, D.-S. ChemMORT: An automatic ADMET optimization platform using deep learning and multi-objective particle swarm optimization. *Brief. Bioinform.* **2024**, *25*, bbae008. [CrossRef]
23. Kronfeld, M.; Planatscher, H.; Zell, A. The EvA2 Optimization Framework. In *Learning and Intelligent Optimization*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 247–250.
24. Ventura, S.; Romero, C.; Zafra, A.; Delgado, J.A.; Hervás, C. JCLEC: A Java framework for evolutionary computation. *Soft Comput.-Fusion Found. Methodol. Appl.* **2008**, *12*, 381–392. [CrossRef]
25. Liefooghe, A.; Basseur, M.; Jourdan, L.; Talbi, E.-G. ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Matsushima, Japan, 5–8 March 2007; pp. 386–400.

26. Bleuler, S.; Laumanns, M.; Thiele, L.; Zitzler, E. PISA-a platform and programming language independent interface for search algorithms. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Faro, Portugal, 8–11 April 2003; pp. 494–508.
27. Wagner, S.; Affenzeller, M. HeuristicLab: A Generic and Extensible. In *Adaptive and Natural Computing Algorithms*; Springer: Vienna, Austria, 2005; pp. 538–541.
28. Van Eck, N.J.; Waltman, L. VOSviewer Manual. 2017. Available online: http://vosviewer.com/download/f-y2z2.pdf (accessed on 12 March 2024).
29. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
30. Zitzler, E.; Laumanns, M.; Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*; TIK-Report; ETH Zurich, Computer Engineering and Networks Laboratory: Zurich, Switzerland, 2001.
31. Deb, K.; Goel, T. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. In *Evolutionary Multi-Criterion Optimization*; EMO 2001; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001; Volume 1993.
32. Deb, K.; Jain, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **2014**, *18*, 577–601. [CrossRef]
33. Jain, H.; Deb, K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **2014**, *18*, 602–622. [CrossRef]
34. He, Z.; Yen, G.G.; Zhang, J. Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **2013**, *18*, 269–285. [CrossRef]
35. Nebro, A.; Durillo, J.; García-Nieto, J.; Coello, C.; Luna, F.; Alba, E. SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In Proceedings of the IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, MCDM, Nashville, TN, USA, 30 March–2 April 2009; pp. 66–73.
36. Fathollahi-Fard, A.M.; Hajiaghaei-Keshteli, M. Red Deer Algorithm (RDA): A new optimization algorithm inspired by Red Deers' mating. In Proceedings of the International Conference on Industrial Engineering, Tehran, Iran, 25–26 January 2016; Volume 12, pp. 331–342.
37. Mirjalili, S.; Mirjalili, S.M.; Lewis, A.; Optimizer, G.W. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61.
38. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [CrossRef]
39. Premkumar, M.; Jangir, P.; Sowmya, R. MOGBO: A new Multi-objective Gradient-Based Optimizer for real-world structural optimization problems. *Knowl.-Based Syst.* **2021**, *218*, 106856. [CrossRef]
40. Dulebenets, M.A. An Adaptive Polyploid Memetic Algorithm for scheduling trucks at a cross-docking terminal. *Inf. Sci.* **2021**, *565*, 390–421. [CrossRef]
41. Wei, Z.; Wang, H.; Wang, S.; Zhang, Z.; Cui, Z.; Wang, F.; Peng, H.; Zhao, J. Many-objective evolutionary algorithm based on parallel distance for handling irregular Pareto fronts. *Swarm Evol. Comput.* **2024**, *86*, 101539. [CrossRef]
42. Xiong, J.; Liu, G.; Gao, Z.; Zhou, C.; Hu, P.; Bao, Q. A many-objective evolutionary algorithm based on learning assessment and mapping guidance of historical superior information. *J. Comput. Des. Eng.* **2024**, *11*, 194–229. [CrossRef]
43. Chu, X.; Ming, F.; Gong, W. Competitive Multitasking for Computational Resource Allocation in Evolutionary Constrained Multi-Objective Optimization. *IEEE Trans. Evol. Comput.* 2024; *Early Access*.
44. Neghină, M.; Dicoiu, A.-I.; Chiş, R.; Florea, A. A competitive new multi-objective optimization genetic algorithm based on apparent front ranking. *Eng. Appl. Artif. Intell.* **2024**, *132*, 107870. [CrossRef]
45. Blank, J.; Deb, K. Pymoo: Multi-Objective Optimization in Python. *IEEE Access* **2020**, *8*, 89497–89509. [CrossRef]
46. Durillo, J.; Nebro, A.; Luna, F.; Dorronsoro, B.; Alba, E.; Teatinos, C. *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*; TECH-REPORT: ITI-2006-10; University of Malaga: Malaga, Spain, 2006.
47. Durillo, J.J.; Nebro, A.J. jMetal: A Java framework for multi-objective optimization. *Adv. Eng. Softw.* **2011**, *42*, 760–771. [CrossRef]
48. Aldana-Martín, J.F.; Durillo, J.J.; Nebro, A.J. Evolver: Meta-optimizing multi-objective metaheuristics. *SoftwareX* **2023**, *24*, 101551. [CrossRef]
49. Nebro, A.J.; Durillo, J.J.; Vergne, M. Redesigning the jMetal Multi-Objective Optimization Framework. In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion '15), Madrid, Spain, 11–15 July 2015; pp. 1093–1100.
50. Calborean, H. Multi-Objective Optimization of Advanced Computer Architectures Using Domain-Knowledge. Ph.D. Thesis, "Lucian Blaga" University of Sibiu, Sibiu, Romania, 2011.
51. Chiș, R. Developing Effective Multi-Objective Optimization Methods for Complex Computing Systems. Ph.D. Thesis, "Lucian Blaga" University of Sibiu, Sibiu, Romania, 2017.
52. Zwart, S.P. The ecological impact of high-performance computing in astrophysics. *Nat. Astron.* **2020**, *4*, 819–822. [CrossRef]
53. Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J.; Saraiva, J. Energy efficiency across programming languages: How do energy, time, and memory relate? In Proceedings of the 10th ACM SIGPLAN International Conference, Vancouver, BC, Canada, 23–24 October 2017.

54. Pereira, R.; Couto, M.; Ribeiro, F.; Rua, R.; Cunha, J.; Fernandes, J.P.; Saraiva, J. Ranking programming languages by energy efficiency. *Sci. Comput. Program.* **2021**, *205*, 102609. [CrossRef]
55. Georgiou, S.; Kechagia, M.; Spinellis, D. Analyzing Programming Languages' Energy Consumption: An Empirical Study. In Proceedings of the 21st Pan-Hellenic Conference on Informatics (PCI '17). Association for Computing Machinery, Larissa, Greece, 28–30 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 1–6.
56. Rashid, M.; Ardito, L.; Torchiano, M. Energy Consumption Analysis of Algorithms Implementations. In Proceedings of the Conference: Symposium on Empirical Software Engineering and Measurement, Beijing, China, 22–23 October 2015.
57. Jamil, M.N.; Kor, A.-L. Analyzing energy consumption of nature-inspired optimization algorithms. *Green Technol. Resil. Sustain.* **2022**, *2*, 1. [CrossRef]
58. Florea, A.; Cofaru, I.; Patrausanu, A.; Cofaru, N.; Fiore, U. Superposition of populations in multi-objective evolutionary optimization of car suspensions. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107026. [CrossRef]
59. Florea, A.; Cofaru, N. Implementing some Evolutionary Computing Methods for Determining the Optimal Parameters in the Turning Process. *Appl. Mech. Mater.* **2015**, *809*, 902–907. [CrossRef]
60. Florea, A.; Gellert, A. Different approaches for solving optimization problems using interactive e-learning tools. *Elearning Softw. Educ.* **2014**, 181.
61. Carlson, T.E.; Heirman, W.; Eeckhout, L. Sniper: Exploringthe the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Seattle, WA, USA, 12–18 November 2011; pp. 1–12.
62. Sharkey, J.; Ponomarev, D.; Ghose, K. *M-sim: A Flexible, Multithreaded Architectural Simulation Environment*; Technical Report; Department of Computer Science, State University of New York at Binghamton: Binghamton, NY, USA, 2005.
63. Vintan, L.; Chis, R.; Ismail, M.; Cotofana, C. Improving Computing Systems Automatic Multi-Objective Optimization through Meta-Optimization. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *35*, 1125–1129. [CrossRef]
64. Bücker, M.; Corliss, G.; Hovland, P.; Naumann, U.; Norris, B. Automatic Differentiation: Applications, Theory, and Implementations. *Lect. Notes Comput. Sci. Eng.* **2006**, *50*.
65. Benítez-Hidalgo, A.; Nebro, A.J.; García-Nieto, J.; Oregi, I.; Del Ser, J. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm Evol. Comput.* **2019**, *51*, 100598. [CrossRef]
66. Uhrig, S.; Shehan, B.; Jahr, R.; Ungerer, T. A Two-Dimensional Superscalar Processor Architecture. In Proceedings of the 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, Athens, Greece, 15–20 November 2009; pp. 608–611.
67. Knowles, J.; Corne, D. On metrics for comparing nondominated sets. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 12–17 May 2002; Volume 1, pp. 711–716.
68. Zitzler, E.; Thiele, L.; Laumanns, M.; Fonseca, C.; da Fonseca, V. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comput.* **2003**, *7*, 117–132. [CrossRef]
69. Zitzler, E.; Thiele, L. Multiobjective optimization using evolutionary algorithms—A comparative case study. In Proceedings of the 5th International Conference on Parallel Problem Solving from Nature (PPSN-V), Amsterdam, The Netherlands, 27–30 September 1998; Eiben, A.E., Back, T., Schoenauer, M., Schwefel, H.P., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 292–301.
70. Zitzler, E. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. Thesis, ETH Zurich, Zurich, Switzerland, 1999.
71. Zhou, A.; Jin, Y.; Zhang, Q.; Sendhoff, B.; Tsang, E. Combining Model-based and Genetics-based Offspring Generation for Multi-objective Optimization Using a Convergence Criterion. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 892–899.
72. Laumanns, M.; Thiele, L.; Zitzler, E. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *Eur. J. Oper. Res.* **2006**, *169*, 932–942. [CrossRef]
73. García-León, A.A.; Dauzère-Pérès, S.; Mati, Y. An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. *Comput. Oper. Res.* **2019**, *108*, 187–200. [CrossRef]
74. Van Veldhuizen, D.; Lamont, G. Evolutionary computation and convergence to a Pareto front. In *Late Breaking Papers at the Genetic Programming 1998 Conference*; Koza, J.R., Ed.; Stanford University Bookstore: Stanford, CA, USA, 1998; pp. 221–228.
75. CoelloCoello, C.; ReyesSierra, M. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In *MICAI 2004 Advances in Artificial Intelligence*; Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 688–697.
76. Shajii, A.; Numanagić, I.; Leighton, A.T.; Greenyer, H.; Amarasinghe, S.; Berger, B. A Python-based optimization framework for high-performance genomics. *bioRxiv* **2020**. [CrossRef]
77. Mondelli, M.L.; Magalhães, T.; Loss, G.; Wilde, M.; Foster, I.; Mattoso, M.; Katz, D.; Barbosa, H.; de Vasconcelos, A.T.R.; Ocaña, K.; et al. BioWorkbench: A high-performance framework for managing and analyzing bioinformatics experiments. *PeerJ* **2018**, *6*, e5551. [CrossRef]
78. Zambrano-Vega, C.; Nebro, A.J.; Aldana-Montes, J.F. MO-Phylogenetics: A phylogenetic inference software tool with multi-objective evolutionary metaheuristics. *Methods Ecol. Evol.* **2016**, *7*, 800–805. [CrossRef]
79. Legato, P.; Mazza, R.M.; Fortino, G. A multi-level simulation-based optimization framework for IoT-enabled elderly care systems. *Simul. Model. Pract. Theory* **2022**, *114*, 102420. [CrossRef]

80. Gillis, M.; Urban, R.; Saif, A.; Kamal, N.; Murphy, M. A simulation–optimization framework for optimizing response strategies to epidemics. *Oper. Res. Perspect.* **2021**, *8*, 100210. [CrossRef]
81. Badakhshan, E.; Ball, P. A simulation-optimization approach for integrating physical and financial flows in a supply chain under economic uncertainty. *Oper. Res. Perspect.* **2023**, *10*, 100270. [CrossRef]