Multiobjective Optimization Analysis for Finding Infrastructure-as-Code Deployment Configurations

Eneko Osaba
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
eneko.osaba@tecnalia.com

Josu Diaz-De-Arcaya
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain

Juncal Alonso
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain

Jesus L. Lobo
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain

Gorka Benguria
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain

Iñaki Etxaniz
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain

ABSTRACT

Multiobjective optimization is a hot topic in the artificial intelligence and operations research communities. The design and development of multiobjective methods is a frequent task for researchers and practitioners. As a result of this vibrant activity, a myriad of techniques have been proposed in the literature to date, demonstrating a significant effectiveness for dealing with situations coming from a wide range of real-world areas. This paper is focused on a multiobjective problem related to optimizing Infrastructure-as-Code deployment configurations. The system implemented for solving this problem has been coined as IaC Optimizer Platform (IOP). Despite the fact that a prototypical version of the IOP has been introduced in the literature before, a deeper analysis focused on the resolution of the problem is needed, in order to determine which is the most appropriate multiobjective method for embedding in the IOP. The main motivation behind the analysis conducted in this work is to enhance the IOP performance as much as possible. This is a crucial aspect of this system, deeming that it will be deployed in a real environment, as it is being developed as part of a H2020 European project. Going deeper, we resort in this paper to nine different evolutionary computation-based multiobjective algorithms. For assessing the quality of the considered solvers, 12 different problem instances have been generated based on real-world settings. Results obtained by each method after 10 independent runs have been compared using Friedman's non-parametric tests. Findings reached from the tests carried out lad to the creation of a multi-algorithm system, capable of applying different techniques according to the user's needs.

ICCCM 2023, August 04–06, 2023, Nagoya, Japan

ACM ISBN 979-8-4007-0773-5/23/08…$15.00
https://doi.org/10.1145/3617733.3617777

CCS CONCEPTS

• **Mathematics of computing - Discrete mathematics - Combinatorics - Combinatorial optimization**; • **Computing methodologies - Artificial intelligence - Search methodologies**;

KEYWORDS

Multiobjective Optimization, Evolutionary Computation, PIACERE, NSGA-II

ACM Reference Format:
Eneko Osaba, Josu Diaz-De-Arcaya, Juncal Alonso, Jesus L. Lobo, Gorka Benguria, and Iñaki Etxaniz. 2023. Multiobjective Optimization Analysis for Finding Infrastructure-as-Code Deployment Configurations. In *2023 The 11th International Conference on Computer and Communications Management (ICCCM 2023), August 04–06, 2023, Nagoya, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3617733.3617777

1 INTRODUCTION

The correct solving of multiobjective problems is a recurrent task in artificial intelligence and operations research fields. The remarkable popularity of this area is clearly justified because Multi-Objective Problems (MOP) are particularly frequent in real-world environments. In a nutshell, the addressing of this kind of problems involves the searching of a set of solutions that provide the optimal equilibrium among all the objectives at hand [1]. In MOP optimization, a solution can be considered as Pareto optimal if there is no other outcome better than it in all the deemed objectives. For this reason, the objectives that compose a MOP must be competitive among them.

A plethora of solvers has been developed in the last decades for facing MOPs. Methods coming from fields such as Evolutionary Computation (EC, [2, 3]) and Swarm Intelligence (SI, [4, 5]) have been especially resorted in the last years. Some of the most well-known methods are the Nondominated Sorting Genetic Algorithm II (NSGA-II, [6]), or the multiobjective evolutionary algorithm based on decomposition (MOEA/D, [7]). Regarding application fields, MOPs have demonstrated to be effective for addressing different situations in areas such as logistics [8], energy [9], industry [10] or engineering [11].

In this paper, we embrace MOP optimization for dealing with a problem related to optimizing Infrastructure-as-Code (IaC, [12]) deployment configurations. More specifically, this paper gravitates

around a system coined as IaC Optimizer Platform (IOP), which objective is to find optimized deployment compositions of the IaC on the most appropriate infrastructural elements that fit some preset requirements and objectives. Among the objectives to optimize, some mutually opposed goals can be found, such as the cost of the whole deployment, and its overall performance.

The IOP has been previously introduced in the literature, in the research conducted in [13]. In that article, the system is preliminary introduced, and its application is superficially shown. In that prototypical version, the IOP resorts to NSGA-II for carrying out the optimization. In the present paper, we took a step forward in the development of the IOP, undertaking a detailed experimentation to determine which is the most suitable multiobjective method to deal with the problem of finding optimized IaC deployment configurations.

For performing this experimentation, we test the behavior of nine different multiobjective EC solvers over 12 instances of the problem. These use cases are heterogeneous enough to assure that the conclusions drawn are significant. Details on the techniques used and instances generated are provided in following sections.

The interest on conducting this study is remarkably high since the IOP is expected to work on a real environment. For this reason, its performance should be as good as possible, and the selection of the optimization algorithms employed should be made after a thorough and rigorous study. Finally, it is noteworthy that the IOP has been implemented as part of a Horizon 2020 EU research project. The main goal of this project, named PIACERE (https://www.piacere-project.eu/) is to develop a complete framework for the development, deployment, and operation of IaC running on cloud continuum.

This work is structured as follows. In the following Section2 we introduce the optimization problem to solve. In Section 3 we focus on describing the IOP and all the algorithms that have been considered for the tests carried out on this work. Section 4 is devoted to the experimentation conducted, while Section 5 finishes this paper with conclusions and further work.

2 PROBLEM STATEMENT AND FUNDAMENTALS

To adequately comprehend the optimization problem at hand, let us introduce an example in this section. First, it should be mentioned that, for properly conducting the optimization process, the IOP counts with an Infrastructural Elements Catalog, or IEC, which is full of elements that the solver can consider in order to select the most optimal deployment configuration. Thus, we introduce here a simplified IEC composed of three kinds of elements, [Storage (ST), Database (DB), Virtual Machine (VM)], having five options for each of these categories.

- ST: [St EU, St Spain, AZ 2, G 2, St Spain, St USA]
- DB: [mysq.GB, db.dyn, postgreSQL.GR, r4.large, m3.med]
- VM: [C1France, C2Europe, m5.large, t2.nano, DS13v2]

Additionally, all elements have some associated attributes, such as the expected performances, the overall availability, provider, or cost. Having said that, the main objective of the IOP is to find the best combination of these elements that comply with the user requirements.

```
optimization opt {
    objectives {
        "cost" => min
        "availability" => max
        "performance" => max
    }
    nonfunctional_requirements {
        req1 "Cost <= 100.0" max 100.0 => "cost";
        req2 "Availability >= 98.0%" min 98.0 => "availability";
        req3 "Region" values "00EU" => "region";
        req4 "elements" => "Storage, DB, VM, VM, VM";
    }
}
```

Figure 1: An excerpt of a DOML example introduced as input in the IOP

In this regard, and for obtaining all the information about the user's needs, the IOP should obtain as input a file written in a modeling language coined as DevSecOps Modelling Language (DOML, [14]). This file includes a specific section in which users introduce their optimization objective and requirements.

For the sake of clarity, we depict in Figure 1 an excerpt of a DOML example. Analyzing this file, we can see how the user wants to find the most optimized configuration, composed of a single ST, one DB and three VMs (depicted in the part elements) optimizing three different objectives: cost, availability, and performance. Furthermore, the user deepens on its requirements, introducing three different ones: the cost of the whole deployment should be less than 100€ , the overall expected availability must be higher than 98%, and all the chosen elements should be deployed in Europe.

Considering these objectives and requirements, and resorting the reduced IEC presented above, the IOP codifies candidate solutions as a list of integer values, each one depicting the index of a specific element. Thus, a tentative solution looks like: [0, 1, 0, 2, 3]: [100, 98.5, 10]. On the one hand, and deeming that the combination that the user wants to deploy is [ST, DB, VM, VM, VM], the solution [0, 1, 0, 2, 3] is equal to [St_Eu, db.sql, C1Italy, m3.small,t4.medium]. On the other hand, [100, 98.5, 10] array represents the cost of the whole deployment (100), the expected availability (98.5) and the overall performance (10).

3 IAC OPTIMIZER PLATFORM: DESCRIPTION AND ALGORITHMS CONSIDERED

When the IOP receives the input DOML, it follows the following process: i) it retrieves the user information from the input file, ii) gathers all the available data from the IEC, iii) models the optimization problem to solve and iv) calculates the optimized IaC deployment configuration, considering the user's objectives and meeting all the introduced requirements.

For carrying out the last of these steps, which is the optimization process, the IOP resorts to a multiobjective solver. As mentioned in the introduction, the prototypical version of the IOP described in [13] employs the well-known NSGA-II for conducting this task. Even so, and despite the fact that the quality of this technique has been proven countless of times in the literature, we must undertake a much more in-depth experimentation to assess if NSGA-II is the most appropriate technique for this context, or if the adoption of another method is more appropriate.

Table 1: Main characteristics of the 12 input DOMLs generated. Regarding the optimizing objectives: c - cost; a - availability; p - performance. Instances optimizing three objectives consider c & p & a.

Instance	Objectives	Elements to deploy			Requirements
		VMs	DBs	STs	
DOML_2_0-4-4	2 (C & P)	0	4	4	
DOML_2_1-1-1	2 (C & A)	1	1	1	Cost < 200€
DOML_2_2-2-2	2 (C & A)	2	2	2	Availability > 96%
DOML_2_4-3-3	2 (C & A)	4	3	3	
DOML_2_5-5-5	2 (C & P)	5	5	5	
DOML_2_6-0-0	2 (C & A)	6	0	0	
DOML_3_0-4-4	3	0	4	4	
DOML_3_1-1-1	3	1	1	1	Cost < 200€
DOML_3_2-2-2	3	2	2	2	Availability > 96%
DOML_3_4-3-3	3	4	3	3	
DOML_3_5-5-5	3	5	5	5	
DOML_3_6-0-0	3	6	0	0	

With this motivation in mind, in this paper we measure the performance of nine different EC based multiobjective techniques. Along with the mentioned NSGA-II, the following methods have been deemed:

- Weighting Achievement Scalarizing Function Genetic Algorithm (WASFGA, [15]).
- Global Weighting Achievement Scalarizing Function Genetic Algorithm (GWASFGA, [16]).
- Many-Objective Metaheuristic Based on the R2 Indicator (MOMBI, [17]).
- Improved Many-Objective Metaheuristic Based on the R2 Indicator (MOMBI2, [18]).
- Multiobjective Cellular Genetic Algorithm (MoCell, [19]).
- S metric selection evolutionary multi-objective optimization algorithm (SMSEMOA, [20]).
- Strength Pareto evolutionary algorithm 2 (SPEA2, [21]).
- Non-dominated Sorting Genetic Algorithm III (NSGA-III, [22]).

All these algorithms have been chosen for three main reasons: i) they are well-known, and they have been previously validated by the scientific community, ii) they can be easily adapted to the discrete problem dealt by the IOP, and iii) they can be implemented through the multiobjective framework embraced by the IOP: jMetal JAVA framework [23].

In the following section, we describe the experimentation carried out, detailing how we have measured the performance of each of these methods.

4 EXPERIMENTATION

In order to properly assess the performance of all the nine multiobjective solvers, an in-depth experimentation has been carried out using 12 different input DOMLs. Each of these input files has been generated with the intention of emulating real situations that the IOP will face once it is deployed in a real environment. Thus, these DOML files present different objectives to optimize, as well as a variable number of elements to deploy. This directly impacts on the complexity of the problem, depending on the number of objectives and the size of the solution to be found. We summarize in Table 1 the main characteristics of the benchmark composed. As can be seen, each instance has been named as DOML_A_x-y-z, where A represents the number of objectives, and x,y,z the number of VMs, DBs and STs to deploy, respectively. For the sake of replicability, the generated benchmark is openly available under demand, or online at [24].

Furthermore, regarding the IEC that composes the problem, it is comprised of a total of 156 elements. Divided by type, the IEC contains 99 Virtual Machines, 24 Databases and 33 Storage elements. All these elements have realistic features, coming from providers such as Amazon, Google, Openstack or Azure. Arguably, the size is the employed catalog is big enough for reaching rigorous conclusions.

Regarding the parameterization employed for each technique, as recommended in studies such as [25], similar parameters have been employed in all the methods developed. Thus, all methods count on a population composed of 50 individuals, while for the crossover and mutation functions, SBX [26] and Polynomial Mutation [27] have been chosen, respectively. For the selection, Binary Tournament [28] mechanism has been utilized. Finally, regarding the maximum number of evaluations, it has been fixed in 2500.

Having said that, we depict in Table 2 the main results obtained for each technique in all the generated instances. Every instance has been run 10 independent times, in order to obtain statistically relevant results. Thus, we represent in Table 2 the average hypervolume value obtained by each method.

At a first glimpse, we can conclude that the best performing methods along the benchmark are the NSGA-II and the NSGA-III. In any case, and following the guidelines provided in [29], the Friedman's non-parametric test for multiple comparisons [30] has been conducted for confirming these preliminary conclusions. For performing this statistical test, the results average obtained by each method has been used. Thus, we depict in Table 3 the results gathered after performing the test through KEEL platform [31].

Analyzing the results provided by the Friedman's test, we can certify that the best performing technique along the whole benchmark is the NSGA-II, which is the one that has reached the best

Table 2: Results obtained by each method in all the generated 12 DOML instances.

Instance	GWASFGA	MoCell	MOMBI	MOMBI2	SMSEMOA	SPEA2	WASFGA	NSGA-II	NSGA-III
DOML_2_0-4-4	0.725	0.891	0.976	0.817	0.966	0.858	0.632	**0.998**	0.907
DOML_2_1-1-1	0.790	0.964	**0.989**	0.597	0.947	0.816	0.523	0.966	0.784
DOML_2_2-2-2	0.796	**0.983**	0.973	0.923	0.922	0.982	0.825	**0.983**	0.973
DOML_2_4-3-3	0.937	0.964	0.976	0.937	0.966	**0.985**	0.973	0.964	0.913
DOML_2_5-5-5	0.546	0.928	0.890	0.824	0.961	0.939	0.670	**0.970**	0.950
DOML_2_6-0-0	0.886	0.980	0.924	0.901	0.966	0.983	0.805	0.995	**0.998**
DOML_3_0-4-4	0.626	0.977	0.969	0.749	0.969	0.953	0.497	0.987	**0.996**
DOML_3_1-1-1	0.718	0.544	0.660	0.648	0.745	0.693	0.610	0.727	**0.750**
DOML_3_2-2-2	0.543	0.727	0.711	0.592	0.734	0.709	**0.841**	0.625	0.756
DOML_3_4-3-3	0.549	0.858	0.963	0.674	0.970	0.965	0.590	0.957	**0.966**
DOML_3_5-5-5	0.513	0.971	0.790	0.822	0.944	0.927	0.517	0.946	**0.968**
DOML_3_6-0-0	0.458	0.940	0.975	0.526	**0.984**	0.972	0.610	0.964	0.944

Table 3: Average Friendman Rankings for each considered algorithm. The less the ranking, the better the performance.

Algorithm	Ranking
GWASFGA	7.9583
MoCell	4.5
MOMBI	4.1667
MOMBI2	7.2083
SMSEMOA	3.2917
SPEA2	4.1667
WASFGA	7.0833
NSGA-II	**3.1667**
NSGA-III	3.4583

ranking. In any case, a deeper analysis of the results shown in Table 2 can lead us to much more interesting and valuable conclusions, taking into account that the IOP will work in a real environment in which its performance should be improved as much as possible.

If we analyze the performance of each technique according to the number of objectives it must optimize, we reach to much more accurate insights. Thus, we have conducted two separate Friendman's non-parametric tests. On the one hand, the first test has been performed employing the outcomes obtained for instances DOML_2_0-4-4, DOML_2_1-1-1, DOML_2_2-2-2, DOML_2_4-3-3, DOML_2_5-5-5 and DOML_2_6-0-0. On the other hand, results got in use cases DOML_3_0-4-4, DOML_3_1-1-1, DOML_3_2-2-2, DOML_3_4-3-3, DOML_3_5-5-5 and DOML_3_6-0-0 have been used for performing the second test. In Table 4, we represent the outcomes of these separated Friendman's non-parametric tests.

The findings that can be drawn by examining the results shown in Table 4 are even more valuable than those described above. Despite NSGA-II is, in overall, the method that performs better for the whole benchmark, if we undertake a separate analysis we see that NSGA-III behaves better for instances where the number of objectives to optimize is three.

As a final conclusion, and after carrying out the experimentation described in this study, we determine that the most appropriate way for implementing the IOP is to convert it into a multi-method system. Thus, the IOP should have a pool composed of two solving algorithms, using each of them according to the user needs: for problems with two optimizing objectives, NSGA-II should be used; while for situations with three objectives, the IOP should resort to NSGA-III.

5 CONCLUSIONS AND FURTHER WORK

This paper has gravitated around a multiobjective problem related to optimizing Infrastructure-as-Code deployment configurations. The system implemented for solving this problem has been coined as IaC Optimizer Platform (IOP), and it has been developed in the context of a European H2020 project. Despite a prototypical version of the IOP has been introduced in the literature before [13], a deeper analysis on the multiobjective solving of the problem was needed, in order to determine which is the most appropriate method for embedding in the IOP.

With this motivation in mind, in this work we have tested the performance of nine different multiobjective EC methods through 12 different instances of the problem. Each use case has been run 10 independent times by each considered method and obtained results have been compared resorting to the Friendman's non-parametric test. The main conclusion drawn is that the IOP should be implemented as a multi-algorithm system, using NSGA-II and NSGA-III depending on whether the number of objectives to be optimized is two or three, respectively.

As future work, the further evolution of the IOP has been planned. The deployment of the findings reached on this work should be implemented, and the behavior of the IOP on real-world environments should be analyzed in a deeper manner. On this direction, we have planned to make a deeper experimentation considering aspects such as noisy objectives [32]. As long-term work, the development of more sophisticated algorithms will be implemented for being embedded in the IOP. Other possible future work is related to extending the conducted tests to problems related to other fields in order to have a clear vision of the performance of selected EC multiobjective algorithms in use cases such as the Job-Shop Scheduling Problem [33] or other problems related to economics [34] or energy

Table 4: Separated Friendman's test for instances optimizing two and three objectives. Rankings for each considered algorithm. The less the ranking, the better the performance.

Two Objectives		Three Objectives	
Algorithm	Ranking	Algorithm	Ranking
GWASFGA	7.9167	GWASFGA	8
MoCell	4.1667	MoCell	4.8333
MOMBI	3.5833	MOMBI	4.75
MOMBI2	7.25	MOMBI2	7.1667
SMSEMOA	4	SMSEMOA	2.5833
SPEA2	3.6667	SPEA2	4.6667
WASFGA	7.3333	WASFGA	6.8333
NSGA-II	**2.3333**	NSGA-II	4
NSGA-III	4.75	NSGA-III	**2.1667**

[35]. Finally, we intend to deal with the same problem resorting to revolutionary paradigms such as quantum computing [36], [37].

ACKNOWLEDGMENTS

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No: 101000162 (PIACERE project).

REFERENCES

[1] Deb, K., & Deb, K. (2013). Multi-objective optimization. In Search methodologies: Introductory tutorials in optimization and decision support techniques (pp. 403-449). Boston, MA: Springer US.
[2] Bäck, T., Fogel, D. B., & Michalewicz, Z. (1997). Handbook of evolutionary computation. Release, 97(1), B1.
[3] Del Ser, J., Osaba, E., Molina, D., Yang, X. S., Salcedo-Sanz, S., Camacho, D., ... & Herrera, F. (2019). Bio-inspired computation: Where we stand and what's next. Swarm and Evolutionary Computation, 48, 220-250.
[4] Coello, C. C. (2006). Evolutionary multi-objective optimization: a historical view of the field. IEEE computational intelligence magazine, 1(1), 28-36.
[5] Pozna, C., Precup, R. E., Horváth, E., & Petriu, E. M. (2022). Hybrid particle filter–particle swarm optimization algorithm and application to fuzzy controlled servo systems. IEEE Transactions on Fuzzy Systems, 30(10), 4286-4297.
[6] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182-197.
[7] Zhang, Q., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation, 11(6), 712-731.
[8] Jayarathna, C. P., Agdas, D., Dawes, L., & Yigitcanlar, T. (2021). Multi-objective optimization for sustainable supply chain and logistics: A review. Sustainability, 13(24), 13617.
[9] Cui, Y., Geng, Z., Zhu, Q., & Han, Y. (2017). Multi-objective optimization methods and application in energy saving. Energy, 125, 681-704.
[10] Aslam, T., & Amos, H. N. (2010, October). Multi-objective optimization for supply chain management: A literature review and new development. In 2010 8th International Conference on Supply Chain Management and Information (pp. 1-8). IEEE.
[11] Bhaskar, V., Gupta, S. K., & Ray, A. K. (2000). Applications of multiobjective optimization in chemical engineering. Reviews in chemical engineering, 16(1), 1-54.
[12] Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. Information and Software Technology, 108, 65-77.
[13] Osaba, E., Diaz-de-Arcaya, J., Orue-Echevarria, L., Alonso, J., Lobo, J. L., Benguria, G., & Etxaniz, I. (2022, July). PIACERE project: description and prototype for optimizing infrastructure as code deployment configurations. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 71-72).
[14] Chiari, M., Nitto, E. D., Mucientes, A. N., & Xiang, B. (2023, January). Developing a New DevOps Modelling Language to Support the Creation of Infrastructure as Code. In Advances in Service-Oriented and Cloud Computing: International Workshops of ESOCC 2022, Wittenberg, Germany, March 22–24, 2022, Revised Selected Papers (pp. 88-93). Cham: Springer Nature Switzerland.
[15] Ruiz, A. B., Saborido, R., & Luque, M. (2015). A preference-based evolutionary algorithm for multiobjective optimization: the weighting achievement scalarizing function genetic algorithm. Journal of Global Optimization, 62, 101-129.
[16] Saborido, R., Ruiz, A. B., & Luque, M. (2017). Global WASF-GA: An evolutionary algorithm in multiobjective optimization to approximate the whole Pareto optimal front. Evolutionary computation, 25(2), 309-349.
[17] Gómez, R. H., & Coello, C. A. C. (2013, June). MOMBI: A new metaheuristic for many-objective optimization based on the R2 indicator. In 2013 IEEE congress on evolutionary computation (pp. 2488-2495). IEEE.
[18] Hernández Gómez, R., & Coello Coello, C. A. (2015, July). Improved metaheuristic based on the R2 indicator for many-objective optimization. In Proceedings of the 2015 annual conference on genetic and evolutionary computation (pp. 679-686).
[19] Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., & Alba, E. (2009). Mocell: A cellular genetic algorithm for multiobjective optimization. International Journal of Intelligent Systems, 24(7), 726-746.
[20] Beume, N., Naujoks, B., & Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research, 181(3), 1653-1669.
[21] Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report, 103.
[22] Deb, K., & Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE transactions on evolutionary computation, 18(4), 577-601.
[23] Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software, 42(10), 760-771.
[24] Osaba, Eneko (2023), "DOML instances for optimizing Infrastructure-as-Code deployment configurations", Mendeley Data, V1, doi: 10.17632/gs5kw9hmz8.1
[25] Osaba, E., Carballedo, R., Diaz, F., Onieva, E., Masegosa, A. D., & Perallos, A. (2018). Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. Neurocomputing, 271, 2-8.
[26] Deb, K., & Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. Complex systems, 9(2), 115-148.
[27] Zeng, G. Q., Chen, J., Li, L. M., Chen, M. R., Wu, L., Dai, Y. X., & Zheng, C. W. (2016). An improved multi-objective population-based extremal optimization algorithm with polynomial mutation. Information Sciences, 330, 49-73.
[28] Blickle, T. (2000). Tournament selection. Evolutionary computation, 1, 181-186.
[29] Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation, 1(1), 3-18.
[30] Sheldon, M. R., Fillyaw, M. J., & Thompson, W. D. (1996). The use and interpretation of the Friedman test in the analysis of ordinal-scale data in repeated measures designs. Physiotherapy Research International, 1(4), 221-228.
[31] Derrac, J., Garcia, S., Sanchez, L., & Herrera, F. (2015). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J. Mult. Valued Logic Soft Comput, 17.
[32] Ryter, R., Hanne, T., & Dornberger, R. (2020). Effects of Noisy Multiobjective Test Functions Applied to Evolutionary Optimization Algorithms. Journal of Advances in Information Technology Vol, 11(3).
[33] Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. ORSA Journal on computing, 3(2), 149-156.
[34] Cornuejols, G., & Tütüncü, R. (2006). Optimization methods in finance (Vol. 5). Cambridge University Press.

[35] Szedlak-Stinean, A. I., Precup, R. E., Petriu, E. M., Roman, R. C., Hedrea, E. L., & Bojan-Dragos, C. A. (2022). Extended Kalman filter and Takagi-Sugeno fuzzy observer for a strip winding system. Expert Systems with Applications, 208, 118215.

[36] Bayerstadler, A., Becquin, G., Binder, J., Botter, T., Ehm, H., Ehmer, T., ... & Winter, F. (2021). Industry quantum computing applications. EPJ Quantum Technology, 8(1), 25.

[37] Osaba, E., Villar-Rodriguez, E., & Oregi, I. (2022). A Systematic Literature Review of Quantum Computing for Routing Problems. IEEE Access.