DECISION SUPPORT FRAMEWORK FOR AUTOMATING THE OPTIMIZATION OF EDGE COMPUTING FEDERATIONS

Antonio F. Rodríguez-Liria
Román Cárdenas
Patricia Arroba
José M. Moya

Laboratorio de Sistemas Integrados (LSI)
CCS–Center for Computational Simulation
Universidad Politécnica de Madrid
ETSI Telecomunicación, Avenida Complutense, 30
Madrid 28040, SPAIN
antonio.rliria@alumnos.upm.es
{r.cardenas,p.arroba,jm.moya}@upm.es

José L. Risco-Martín

Dpt. of Computer Architecture and Automation
Universidad Complutense de Madrid
C/ Profesor José García Santesmases, 9
Madrid 28040, SPAIN
jlrisco@ucm.es

Gabriel Wainer

Dpt. of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa ON K1S 5B6, CANADA
gwainer@sce.carleton.ca

ABSTRACT

Modern IoT applications present demanding Quality of Service (QoS) requirements that can no longer rely on a centralized solution to offload compute-intensive tasks. Edge Computing tackles these limitations by extending the Cloud to the edge of the network. However, the distributed and complex nature of Edge infrastructures introduces new technical challenges that must be addressed to develop a successful solution. Modeling, Simulation, and Optimization (M&S&O) tools can aid in the automatic design of modern Edge deployments. Here we present a decision support module that allows practitioners to automatically improve aspects of their Edge infrastructure by applying different optimization algorithms, such as simulated annealing, tabu search, and stochastic hill climbing. Furthermore, the structure of this module allows us to extend the available optimization methods if needed. We illustrate how this tool can help us to improve energy consumption policies of Edge Computing facilities and reduce their operational costs.

Keywords: Edge Computing, IoT, MBSE, Optimization.

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

1 INTRODUCTION AND RELATED WORK

The Internet of Things (IoT) has been gaining prominence over the past few years, with considerable growth in its market from 2014 to the present (Dahlqvist et al. 2019). Furthermore, forecasts suggest that this growth will continue to increase in the future (Lueth 2020). In turn, IoT applications have progressively increased the complexity of their ecosystems, sharing and self-managing their resources autonomously to achieve a common goal in a coordinated manner. However, IoT devices often have intrinsic limitations in terms of storage and processing capacity. Thus, compute-intensive applications and massive data storage require additional infrastructure for effective real-time processing of large volumes of information whose sources are geographically distributed. Traditionally, Cloud infrastructures have provided support to complement IoT services (Stergiou et al. 2018). Some IoT applications have stringent Quality of Service (QoS) requirements such as low latency or high mobility. For these time-critical applications, a centralized data center approach has some limitations. Edge Computing emerges as a solution to these limitations, moving data processing to the edge of the network (Shi et al. 2016). Computing resources are decentralized and distributed geographically closer to devices to reduce latency and network costs (Pan and McElhannon 2018) significantly. Although there is no consensus on the location of Edge Computing yet, we follow the Internet Service Providers (ISPs)' point of view, which devise Edge Computing as a set of distributed computing nodes from the access networks to the public Cloud (Cisco 2022). Edge Data Centers (EDCs) provide real-time data processing and basic analytics, while the Cloud support big data applications and data warehousing. Figure 1 shows a diagram of this Edge Computing architecture.

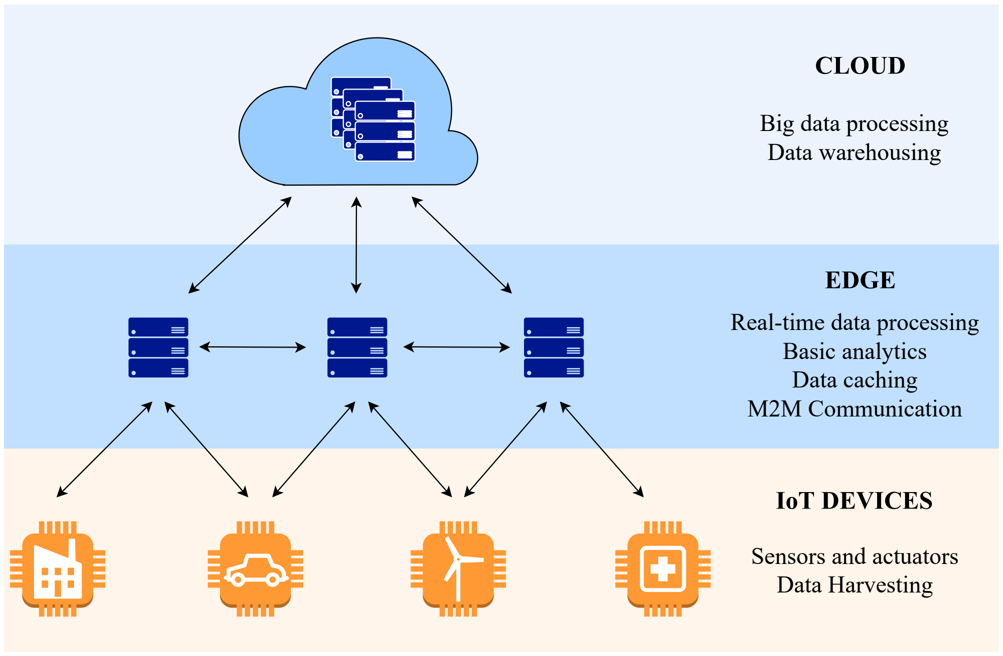


Figure 1: Diagram of the Edge Computing infrastructure.

The current challenge is to design fully functional Edge infrastructures that guarantee the exchange of data between devices and EDCs for computation offloading while decreasing the operational costs and increasing the infrastructure's energy efficiency.

Edge Computing infrastructure Modeling and Simulation (M&S) tools enable the analysis of the operation of these systems prior to their implementation. Modelers can simulate different scenarios to assess how an Edge solution would behave under a given set of conditions. Integrating M&S practices in developing complex systems such as Edge Computing infrastructures allows for finding suitable solutions in less time while

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

minimizing investment costs. Currently, there are numerous software tools for modeling and simulation of IoT and Edge Computing scenarios (e.g., iFogSim (Gupta et al. 2017) or EdgeCloudSim (Sonmez, Ozgovde, and Ersoy 2017)). The Mercury Modeling, Simulation, and Optimization (M&S&O) framework (Cárdenas et al. 2020) allows analyzing federated Edge Computing infrastructures. However, none of these frameworks integrate optimization tools to automatically improve the performance of the scenario based on the results obtained by the simulation and the system requirements. Instead, these changes must be made manually, with the participation of an active observer of the system's operation.

Considering these facts, the aim of this research is the definition and implementation of an optimization tool that assists in the process of designing Edge Computing infrastructures, applying optimization algorithms of different profiles. Mathematical optimization uses system information to obtain the best solution considering specific constraints that must be satisfied. Usually, the objective function or some constraints cannot be handled by classic optimization approaches. Instead, other techniques based on heuristic methods can reach near-optimal solutions in a reasonable amount of time. Among these techniques, metaheuristic algorithms work particularly well, since they can find a sufficiently good system configuration with limited computing resources in a reasonable time frame (Yang 2011). As another advantage, metaheuristics are not problem-specific. Instead, they treat the problem as a black box. Thus, we can apply them to optimize complex systems with numerous degrees of freedom and non-linear behavior. These particularities make metaheuristic algorithms great candidates for optimizing Edge Computing scenarios. We are interested in experimenting with simulated annealing (Delahaye, Chaimatanan, and Mongeau 2019), tabu search (Prajapati, Jain, and Chouhan 2020), and stochastic hill climbing (Mondal, Dasgupta, and Dutta 2012) as examples of metaheuristic algorithms that can help in the complex system development process. These methods are a good starting point because they present a fair compromise between effectiveness and simplicity of implementation. Based on these ideas, we researched how to conduct these studies, and implemented a decision support module for the Mercury M&S&O framework. This tool integrates different optimization algorithms for automating the design and dimensioning of Edge Computing infrastructures. The decision support module applies different optimization techniques and it is flexible enough to solve problems related to system requirements of different natures. We also present a use case in the area of smart cities to illustrate how this optimization tool can contribute to providing a more sustainable infrastructure. Finally, we discuss the performance of different optimization techniques for the proposed case study. The presented decision support system is publicly available on the repository for Mercury (Cárdenas 2023)

The remainder of this paper is organized as follows. Section 2 summarizes the architecture of Mercury, the M&S&O framework for Edge Computing for which the decision support framework is designed. Section 3 elaborates on the main elements of the implemented decision support framework. We define a case study scenario and illustrate how the decision support framework works in Section 4. Finally, we draw conclusions and propose future work in Section 5.

2 BACKGROUND: MERCURY

Mercury is an M&S&O framework for federated Edge Computing infrastructures. Mercury provides a fine-grained level of detail of all the entities comprising the Edge Computing scenario to assist in the decision-making process when designing and operating Edge Computing facilities. This tool is built on top of the xDEVS simulation engine (Risco-Martín et al. 2023), a framework for describing systems using the Discrete EVent System specification (DEVS) formalism (Zeigler, Muzy, and Kofman 2018). Figure 2 represents the principal elements of Mercury. The **clients layer** represents the end users that utilize Edge Computing resources for their applications. Clients are usually embedded systems with intrinsically limited processing and energy resources. These embedded systems offload compute-intensive tasks to one of the EDCs comprising the **EDCs layer**. To offload requests to an EDC, clients and EDCs must be connected to the same network. The **gateways layer** contains all the nodes installed by the ISP that connect clients to the rest of

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

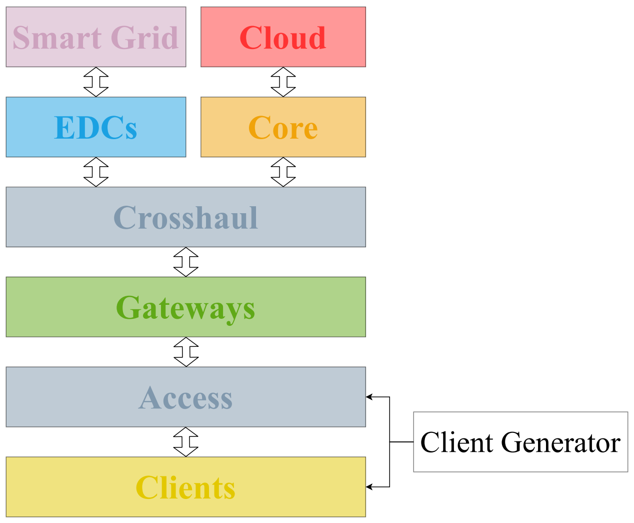


Figure 2: Schematic of Mercury.

the scenario. The **core layer** models all the infrastructure management functions performed by the ISP that owns the network. In particular, two of these functions are modeled: the Access and Mobility Management Function (AMF), which processes new connection requests from clients to ensure that they have permission to connect and keeps track of which client is connected to which gateway, and the Software-Defined Network Controller (SDNC), which is in charge of routing the incoming computation offloading requests to the most suitable EDC of the federation. Additionally, the core network layer acts as an intermediary between the **Cloud layer** and the rest of the scenario. Cloud facilities serve as a backup in case the Edge Computing federation gets congested and cannot provide service to all the clients in the scenario. The **crosshaul and access layers** model communication networks are used by the rest of the elements to communicate with each other. Alternatively, the **client generator** module manages the creation and removal of clients during a simulation.

Mercury implements a novel smart grid-awareness approach for improving the carbon footprint of Edge Computing facilities, presented in Figure 3. In Mercury's **smart grid layer**, an energy provider supplies

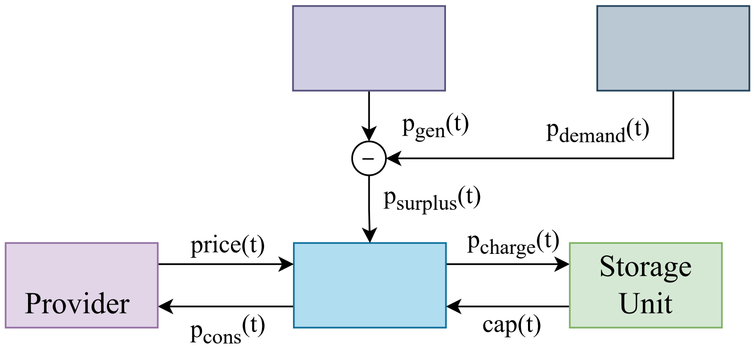


Figure 3: Smart grid-aware model in Mercury.

electricity to the EDCs at a fluctuating price $price(t)$ $\mathrm{\$MWh^{-1}}$. Each EDC contains a smart grid consumer. A smart grid consumer consists of a set of energy sources (e.g., solar panels) that generate $p_{gen}(t)$ W, a power storage unit (e.g., a battery) with $cap(t)$ Wh, and an energy storage controller. The storage controller decides when to charge or discharge the energy storage unit of the consumer depending on the current energy price, energy generation, and energy demand to minimize the power consumption of the EDC, $p_{cons}(t)$. This model

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

allows us to explore the coordinated management of power generation, storage, cooling, and computing capabilities for reducing the energy consumption of the Edge federation. For more details regarding the energy storage controller management workflow, refer to our previous work (Cárdenas et al. 2023).

Currently, Mercury only supports one energy storage controller policy. This policy allows charging or discharging the power storage unit depending on the energy price offered by the energy provider. In this policy, we must specify two different electricity price thresholds. The *maximum charging price*, $\mathbf{price_{charge}}$, corresponds to the maximum electricity price (in $\$\,\mathrm{MWh}^{-1}$) that the consumer (in this case, the EDC) is willing to pay for charging its energy storage unit. On the other hand, the *minimum discharging price*, $\mathbf{price_{charge}}$, specifies the minimum electricity price (in $\$\,\mathrm{MWh}^{-1}$) for the energy storage controller to subtract energy from the storage unit.

3 DECISION SUPPORT FRAMEWORK

This Section presents the decision support system implemented for Mercury. Figure 4 shows a schematic of the implemented framework and the relationships with other modules of the M&S&O tool. First, we

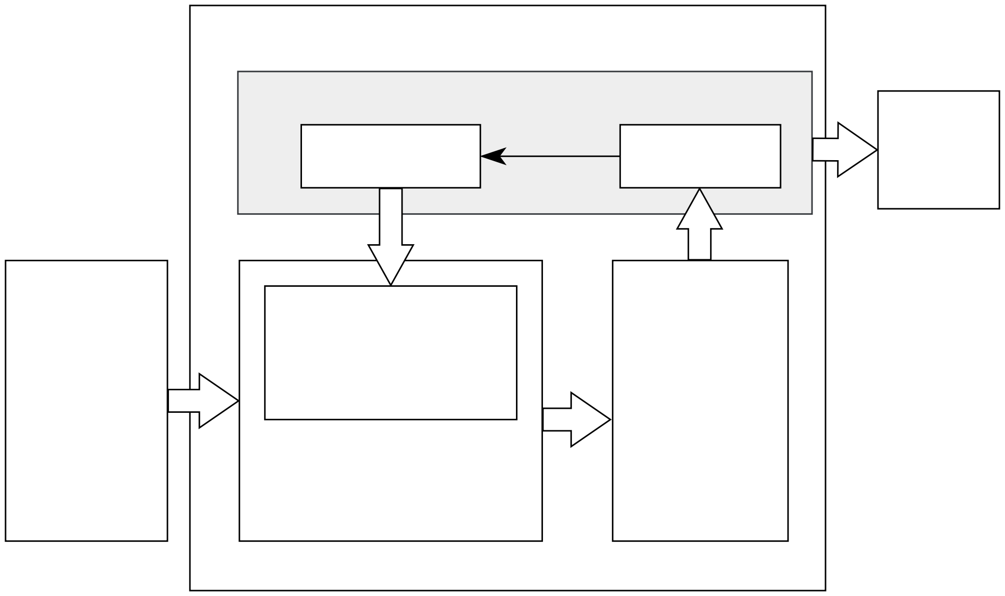


Figure 4: Schematic of the decision support system.

must define a set of system requirements our solution must accomplish (e.g., a certain level of QoS). These requirements comprise the **system specification**. Then, we describe the initial Edge Computing scenario configuration. In Mercury, all model aspects can be configured using a JavaScript Object Notation (JSON) file. Mercury and the xDEVS engine work jointly to simulate the described scenario. Simulation results are stored in a set of Comma-Separated Values (CSV) files with simulation traces. These traces contain information regarding delays perceived by clients, the bandwidth and binary rates of communication links, and the power consumption of the Edge Computing federation, among others. Next, the chosen **optimizer** evaluates how good the simulated scenario is by executing a scenario **cost function**. The optimizer allows modelers and practitioners to implement this cost function according to their specific system requirements. Then, the optimizer iteratively generates alternative scenario configurations according to a given **new state generation policy**, simulates the new setup, and evaluates how good the new configuration is according to the cost function. The state generation policy can also be configured depending on the part of the system

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

under study. Users can select the number of iterations the optimizer will execute. Eventually, the decision support system returns the configuration of the best solution obtained by the framework.

The logic for the cost function and the new state generation is encapsulated under the `Optimizer` Abstract Base Class (ABC). This class is a standard interface that any optimization methodology must implement to be compatible with the proposed decision support system. Figure 5 shows a simplified Unified Modeling Language (UML) class diagram of the `Optimizer` class. In this diagram, `S` represents the data type used for representing a scenario configuration. Alternatively, `C` corresponds to the data type used to represent the cost of a scenario. Usually, `C` is set to `float` (i.e., a floating-point decimal number). However, the `Optimizer` class supports any data type that can be ordered. This feature allows practitioners to implement complex cost functions depending on their requirements (e.g., multi-objective functions with compound cost functions). Any class that inherits the `Optimizer` class presents the following attributes:

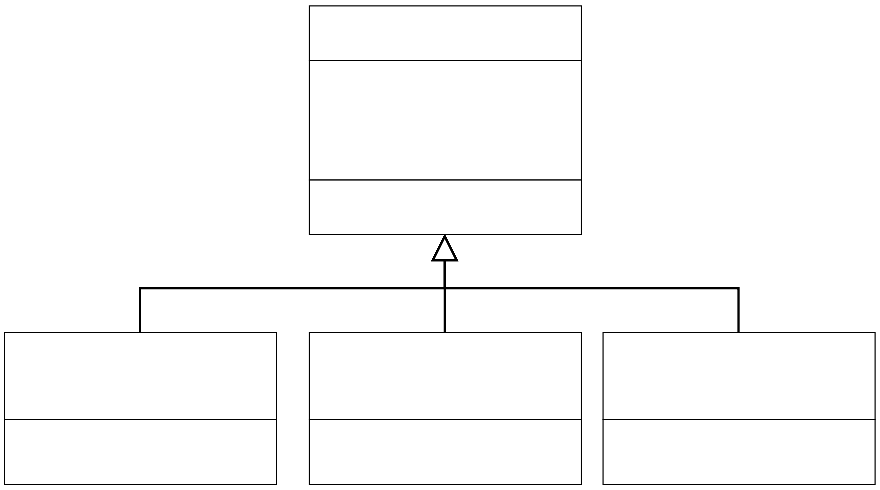


Figure 5: UML class diagram of the optimizer class.

- `state`: current system configuration candidate. This attribute is modified every time the optimizer moves to a new state.
- `best_state`: best scenario found so far.
- `cost_history`: list of previous candidate scenarios' costs. The optimizer appends to this list the associated cost of every candidate scenario.
- `preconditions`: list of functions that, given a system configuration, determine whether or not the configuration is valid before simulating the scenario. Usually, simulating a scenario of a complex system takes a considerable amount of time. The preconditions list is a convenient mechanism to detect *invalid* setups before executing a simulation, thus saving time.

Alternatively, the `Optimizer` class presents the abstract method `run`. This method is in charge of executing the optimization loop. However, the particularities of its logic depend on each implementation of the `Optimizer` class. This is how the ABC represents the tool's high-level structure without specifying the optimization technique nor the part of the infrastructure to be optimized.

The proposed decision support tool of Mercury supports three optimization techniques: **simulated annealing** (Delahaye, Chaimatanan, and Mongeau 2019), **tabu search** (Prajapati, Jain, and Chouhan 2020), and **stochastic hill climbing** (Mondal, Dasgupta, and Dutta 2012). The `SimAnnealOptimizer` class implements the `run` method for performing simulated annealing optimizations. This algorithm is based

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

on the physical annealing process, which aims to bring an element to its minimum energy point by increasing its temperature and gradually decreasing it. The lowest energy configuration of the element is reached only if the maximum temperature is sufficiently high and the cooling is sufficiently slow. In simulated annealing, the cost function corresponds to the energy of the system. Simulated annealing helps optimize complex systems whose search space is nonlinear and has numerous local minima. In contrast, the `TabuSearchOptimizer` performs the tabu search optimization algorithm. Tabu search is a local search-based metaheuristic method. It is a modification of the iterative top-down local search method, used to search in each iteration for a solution that improves the objective function from among a set of alternatives obtained from small displacements from a given state. However, tabu search incorporates the best state of each iteration into a *taboo* list to avoid revisiting it. Finally, `StHillClimbOptimizer` implements the stochastic hill climbing algorithm. This method combines the local search procedure with stochastic decision-making. This algorithm is a simple loop that, starting from an initial state, moves to states with a lower cost function value until a minimum is found, randomly selecting the direction depending on the amount of progress involved. We used the Solid Python package to aid us in implementing the logic of these optimization algorithms (Soni et al. 2017). Each optimization algorithm presents two additional abstract methods:

- `cost`: a cost function used to evaluate candidate solutions. Practitioners must define a cost function that correctly evaluates system configurations according to their particular system requirements.
- `move`: a function that generates a new state from the previous candidate solution. Practitioners must define a state generation function that i) modifies those parts of the system under study and ii) aligns with the nature of the optimization algorithm used. For instance, simulated annealing supports wider steps to avoid local optima, while stochastic hill climbing depends on the cost history.

Due to the particularities of the tabu search algorithm, the `TabuSearchOptimizer` class is the only implementation that supports parallel execution. The tabu search algorithm generates *N* alternative configurations in each iteration, called *neighbors*. These neighbors can then be simulated and evaluated in parallel, reducing the execution time. Next, we present a case study to illustrate how the presented decision support framework can reduce the energy consumption of the Edge infrastructure.

4 CASE STUDY

Here we illustrate the workflow of the decision support tool for optimizing the energy cost of Edge Computing infrastructures. In particular, we optimized the policy of the energy storage controller modules of Mercury's smart grid layer. We used the three implemented optimization algorithms and compared the results obtained. We executed the experiments on a Windows 10 laptop with an Intel Core i5 processor, 16 GB memory, Python 3.9, and the Spyder 5.3.2 Integrated Development Environment (IDE).

The setup corresponds to the San Francisco bay area, where clients of the Edge Computing infrastructure are taxis with an onboard Advanced Driver Assistance System (ADAS). These taxis periodically send computation offloading requests to the EDCs in the scenario. Figure 6 represents the proposed scenario. The scenario comprises three EDCs (represented by big dots) and 19 gateways (displayed as triangles). The color of the gateways corresponds to their closest EDC. The number of clients and their location emulate real mobility traces measured in the city of San Francisco on June 6th, 2008 (Piorkowski et al. 2009). Taxi locations are depicted as small dots. Their color corresponds to their closest gateway.

EDCs incorporate batteries with a maximum capacity of 1 kWh and a maximum charge and discharge rate of 200 W. Additionally, each EDC has Photovoltaic (PV) panels to reduce their overall power consumption. The PV panels' power generation profile was generated using the PVGIS-NSRDB database according to the

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

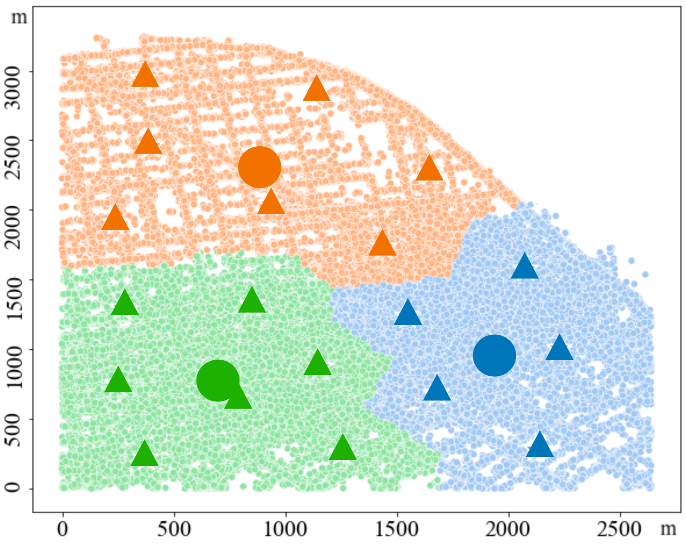


Figure 6: Scenario under study: San Francisco bay area.

location of the EDCs. The electricity price offered by the energy provider corresponds to the average hourly wholesale energy price in California in 2017. For every optimization algorithm, we set the `cost` function as the total cost (in \$) of the energy consumed by the EDCs in the scenario. Even though the `move` function depends on the optimization algorithm, all the implementations explored alternative values for $price_{charge}$ and $price_{discharge}$ in each EDC. For every experiment, we set the initial value of these parameters to 20 and 37 $\$\,\mathrm{MWh}^{-1}$, respectively.

4.1 Optimization Results

First, we configured the presented decision support system to optimize the scenario under study using *simulated annealing*. We executed 100 iterations of this algorithm. We set the maximum and minimum temperatures to 25,000 and 2.5, respectively. These are the default configuration parameters of the SolidPy Python package. In future work, we plan to study these parameters' effects on the algorithm's performance. Figure 7 shows the cost of the scenario for each iteration. The cost of the best scenario is shown in blue. The

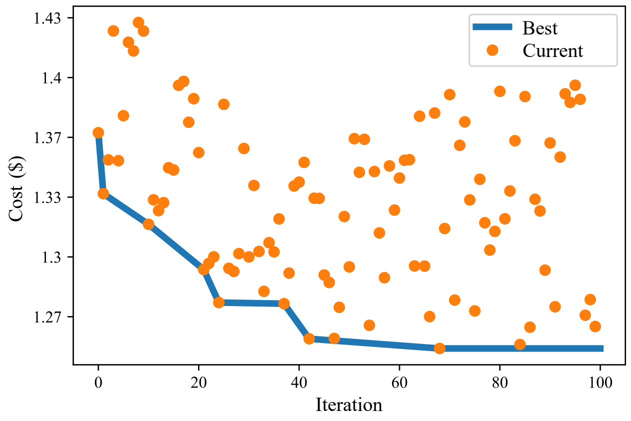


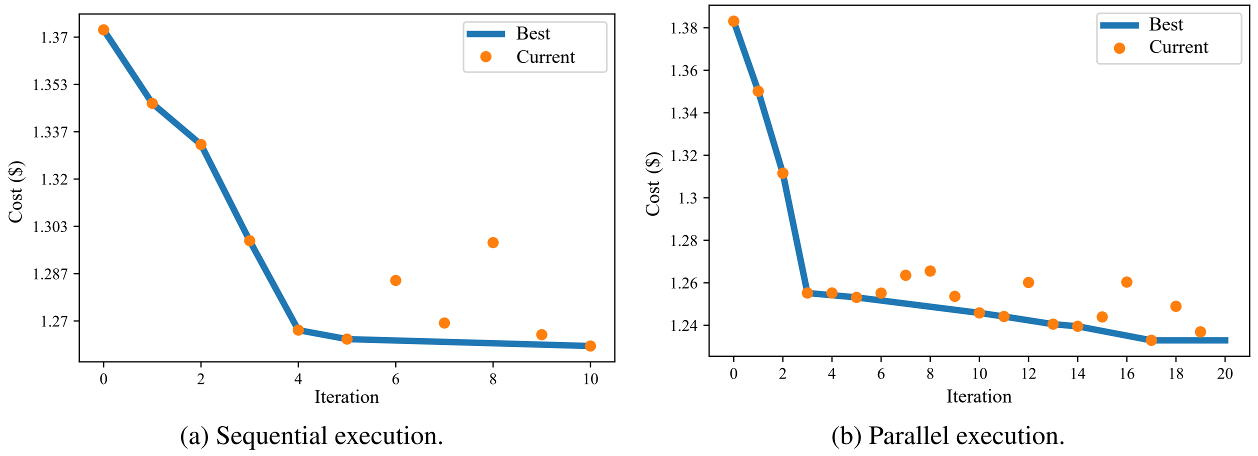
Figure 7: Cost history with simulated annealing.

total execution time was 121:25:59, (i.e., 1 hour, 17 minutes, and 2 seconds per iteration). The best scenario

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

obtained a total cost of 1.25 \$. The cost function results of each iteration are widely distributed across the graph. This is because simulated annealing is designed to modify the scenario under study considerably to avoid potential local minima. This algorithm can broadly evaluate the search space in question in a reduced number of iterations.

Next, we optimized the same scenario using the tabu search algorithm. The length of the tabu table was set to 50. For the tabu search algorithm, we used sequential and parallel execution. We performed 10 iterations of this algorithm in sequential mode with 10 neighbors per iteration. In total, it simulated 100 scenarios. On the other hand, parallel execution required significantly less time. Thus, we performed 20 iterations within a reasonable time span, resulting in 200 different scenarios. Figure 8 show the obtained results. Note that, in



(a) Sequential execution.

(b) Parallel execution.

Figure 8: Cost history with tabu search.

this case, we only display the cost of the best neighbor of every iteration. As a result, the cost function is less scattered compared to simulated annealing. The sequential optimization took 131 hours, 32 minutes, and 15 seconds to complete (i.e., 1 hour, 18 minutes, and 55 seconds per neighbor), and obtained an overall cost of 1.26 \$ (i.e., slightly higher than simulated annealing). In contrast, the parallel version performed two times more iterations in 85 hours, 54 minutes, and 41 seconds (i.e., 25 minutes and 46 seconds per neighbor). Thus, the parallel version of tabu search achieved a speedup of 3.06 over the sequential implementation. Additionally, parallel tabu search obtained the best results given the cost function provided (i.e., a total energy cost of 1.23 \$).

Finally, we evaluated the performance of the stochastic hill climbing algorithm. In this case, the new state generation function consists of slightly increasing or diminishing the values under study for each EDC. Figure 9 shows the results obtained with this optimizer. The resulting graph clearly shows the gradual evolution of the cost function from the initial state to the solution obtained. However, the stochastic hill climbing algorithm did not reach an optimum as good as the other methods. The trend of the best cost curve led us to think that this algorithm would eventually find a solution similar to the other algorithms if we performed more iterations. However, this would considerably increase total execution time, and there is still the possibility that it will not reach the global optimum.

Table 1 compares the obtained results for all the optimization algorithms under study. Stochastic hill climbing showed the worst results, mainly because it required more iterations to reach the global optimum. With stochastic hill climbing, the evolution of the objective functions is so gradual that it becomes necessary to extend the number of iterations to avoid local optima, with the consequent increase of the optimizer execution time. If we only consider sequential execution, the simulated annealing algorithm is a very useful alternative that significantly extends the search space with fewer steps. However, multiprocessing features

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

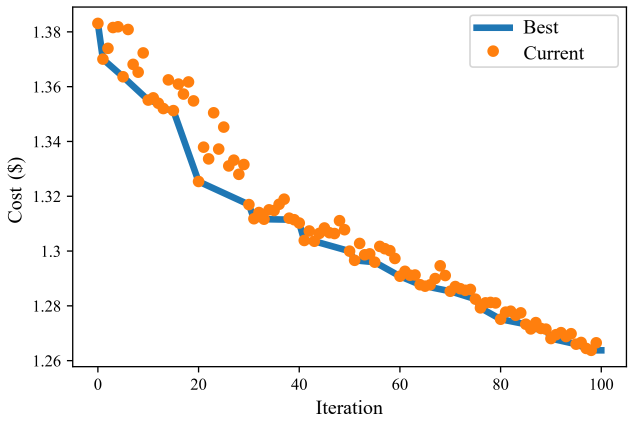


Figure 9: Cost history with stochastic hill climbing.

Table 1: Comparison of the optimization methods under study.

	Sim. annealing	Tabu search (seq.)	Tabu search (par.)	St. hill climbing
Scenarios	100	100	200	100
Time (h:mm:ss)	128:22:39	131:32:15	85:54:41	139:29:52
Best cost ($)	1.25	1.26	1.23	1.27
$\mathbf{price_{charge}}$ ($\$\,\mathrm{MWh}^{-1}$)	EDC1: 21 EDC2: 28 EDC3: 15	EDC1: 22 EDC2: 27 EDC3: 16	EDC1: 22 EDC2: 22 EDC3: 20	EDC1: 30 EDC2: 30 EDC3: 25
$\mathbf{price_{discharge}}$ ($\$\,\mathrm{MWh}^{-1}$)	EDC1: 32 EDC2: 54 EDC3: 35	EDC1: 28 EDC2: 54 EDC3: 25	EDC1: 29 EDC2: 45 EDC3: 36	EDC1: 35 EDC2: 50 EDC3: 30

provide an enormous advantage to the tabu search method with parallel neighbor evaluation, allowing us to obtain better results in less time since it becomes feasible to increase the number of iterations. As a general rule, optimization methods that can run in parallel will potentially fit better for complex systems, as computing the cost function of a given state involves performing a time-costly simulation. The speedup obtained by these tools will be close to the number of cores of the machine performing the optimization process. Therefore, in future works, we will prioritize optimization algorithms that can run in parallel over other alternatives.

5 CONCLUSIONS AND FUTURE WORK

IoT applications with demanding QoS requirements can no longer rely on a centralized approach to offload compute-intensive tasks. Edge Computing arises as a new computation offloading paradigm to tackle these limitations. However, we must design fully functional Edge infrastructures guaranteeing the demanded QoS while reducing development expenses and increasing energy efficiency. In this context, M&S&O tools can significantly aid in the design, implementation, and operation of modern Edge solutions.

In this paper, we presented a decision support tool for Mercury, an M&S&O framework for Edge Computing federations. This tool applies stochastic, metaheuristic, and iterative optimization algorithms to automatically improve the configuration of Edge Computing scenarios according to their system requirements. The

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

structure of the tool, based on abstract classes, allows for solving different problems related to Edge Computing infrastructure requirements. We presented a use case that aims to reduce the cost of energy consumption of the infrastructure using all the optimization algorithms implemented in the tool to date. Namely, our tool currently supports simulated annealing, tabu search, and stochastic hill climbing. Although these three algorithms obtained similar results, the parallel neighbor evaluation for the tabu search method significantly reduced the execution times, making it the best solution both for the results provided and the time spent. The presented decision support system is publicly available on Mercury's official repository (Cárdenas 2023). As future work, we are currently studying the effectiveness of the presented tool for multi-objective optimization problems that allow us to optimize several aspects of the infrastructure while simultaneously looking for a consensus among them. We also plan to add new optimization algorithms that allow simple implementations for parallel execution, such as genetic algorithms.

ACKNOWLEDGEMENTS

We would like to thank Huawei Spain for partially funding this work under the research grant ESC.JRS.001. This work has also been partially supported by the Spanish State Research Agency (Agencia Estatal de Investigación), under grant PID2019-110866RB-I00/AEI/10.13039/501100011033, from the Spanish Ministry of Science and Innovation. The research was partially supported by NSERC (Canada).

REFERENCES

Cisco 2022. "What is Edge Computing?". Technical report, Cisco Systems, Inc. Available via https://www.cisco.com/c/en/us/solutions/computing/what-is-edge-computing.html. Accessed Jan. 17, 2023.

Cárdenas, R. 2023. "Mercury M&S&O Framework for Fog Computing". Available via https://github.com/greenlsi/mercury_mso_framework. Accessed Mar. 14, 2023.

Cárdenas, R., P. Arroba, R. Blanco, P. Malagón, J. L. Risco-Martín, and J. M. Moya. 2020. "Mercury: A modeling, simulation, and optimization framework for data stream-oriented IoT applications". *Simulation Modelling Practice and Theory* vol. 101, pp. 102037.

Cárdenas, R., P. Arroba, J. L. Risco-Martín, and J. M. Moya. 2023. "Modeling and simulation of smart grid-aware edge computing federations". *Cluster Computing* vol. 26, pp. 719—743.

Dahlqvist, F., M. Patel, A. Rajko, and J. Shulman. 2019, July. "Growing opportunities in the Internet of Things". white paper, McKinsey & Company, New York, USA.

Delahaye, D., S. Chaimatanan, and M. Mongeau. 2019. "Simulated annealing: From basics to applications". In *Handbook of metaheuristics*, pp. 1–35. Springer.

Gupta, H., A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. 2017. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments". *Software: Practice and Experience* vol. 47 (9), pp. 1275–1296.

Lueth, K. L. 2020, November. "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time". white paper, IoT Analytics GmbH, Hamburg, Germany.

Mondal, B., K. Dasgupta, and P. Dutta. 2012. "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach". *Procedia Technology* vol. 4, pp. 783–789.

Pan, J., and J. McElhannon. 2018, February. "Future Edge Cloud and Edge Computing for Internet of Things Applications". *IEEE Internet of Things Journal* vol. 5 (1), pp. 439–449.

Piorkowski, M., N. Sarafijanovic-Djukic, and M. Grossglauser. 2009. "A parsimonious model of mobile partitioned networks with clustering". In *2009 First International Communication Systems and Networks and Workshops*, pp. 1–10. Institute of Electrical and Electronics Engineers, Inc.

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.

Prajapati, V. K., M. Jain, and L. Chouhan. 2020. "Tabu search algorithm (TSA): A comprehensive survey". In *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, pp. 1–8. IEEE.

Risco-Martín, J. L., S. Mittal, K. Henares, R. Cardenas, and P. Arroba. 2023. "xDEVS: A toolkit for interoperable modeling and simulation of formal discrete event systems". *Software: Practice and Experience* vol. 53 (3), pp. 719–743.

Shi, W., J. Cao, Q. Zhang, Y. Li, and L. Xu. 2016. "Edge Computing: Vision and Challenges". *IEEE Internet of Things Journal* vol. 3 (5), pp. 637–646.

Soni, D., C. Clauss, F. Dimitrovski, and E. Ercan. 2017. "Solid: a comprehensive gradient-free optimization framework written in Python". [Accessed on: January 25, 2023] https://github.com/100/Solid.

Sonmez, C., A. Ozgovde, and C. Ersoy. 2017. "EdgeCloudSim: An environment for performance evaluation of Edge Computing systems". In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 39–44. IEEE.

Stergiou, C., K. E. Psannis, B.-G. Kim, and B. Gupta. 2018. "Secure integration of IoT and Cloud Computing". *Future Generation Computer Systems* vol. 78, pp. 964–975.

Yang, X.-S. 2011. "Metaheuristic optimization: algorithm analysis and open problems". In *International Symposium on Experimental Algorithms*, pp. 21–32. Springer.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of modeling and simulation: discrete event & iterative system computational foundations*. Academic press.

AUTHOR BIOGRAPHIES

ANTONIO F. RODRÍGUEZ-LIRIA received the M.Sc Degree in Electronics Engineering in 2023 from Universidad Politécnica de Madrid (UPM), Spain. His research interests are aligned with next-generation 5G infrastructures for smart cities. He can be reached at antonio.rliria@alumnos.upm.es.

ROMÁN CÁRDENAS received the M.Sc. Degree in Telecommunication Engineering in 2019 from Universidad Politécnica de Madrid (UPM), Spain, where he pursues a Ph.D. in Electronic Systems Engineering in Cotutelle with Carleton University (CU). His research interests include modeling and simulation with applications in the IoT domain. His email address is r.cardenas@upm.es.

PATRICIA ARROBA is an Assistant Professor at Universidad Politécnica de Madrid (UPM), Spain. She received her Ph.D. Degree in Telecommunication Engineering from UPM in 2017. Her research interests include energy and thermal-aware modeling and optimization of data centers. She can be reached at p.arroba@upm.es.

JOSÉ M. MOYA is an Associate Professor at Universidad Politécnica de Madrid (UPM). He received his Ph.D. degree in Telecommunication Engineering from UPM in 2003. His research interests include proactive and reactive thermal-aware optimization of data centers. His email address is jm.moya@upm.es.

JOSÉ L. RISCO-MARTÍN received his Ph.D. from Universidad Complutense de Madrid (UCM), Spain, where he currently is a Full Professor in the Department of Computer Architecture and Automation. His research interests include computer-aided design and modeling, simulation, and optimization of complex systems. His email address is jlrisco@ucm.es.

GABRIEL WAINER received the Ph.D. degree from Université d'Aix-Marseille III, France. He is a Full Professor at Carleton University (CU). His current research interests are related to modeling methodologies and tools, parallel/distributed simulation, and real-time systems. His email address is gwainer@sce.carleton.ca.

Authorized licensed use limited to: Carleton University. Downloaded on August 31,2024 at 14:09:39 UTC from IEEE Xplore. Restrictions apply.