



OSIRES user manual (Open Source Iterative Resolution for Event Simulation)

Manual do usuário para o projeto OSIRES (Open Source Iterative Resolution for Event Simulation - Resolução Iterativa de Código Aberto para Simulação de Eventos) ver. 02.

Authors/Autores

Prof. Dr. Wilson Trigueiro de Sousa Júnior* (gestor do projeto - wilson.trigueiro@ufsj.edu.br)

Alex de Andrade Soares (programmer)

Luís Arthur de Assis Moraes (programmer)

* correspondent author

Summary

1. The OSIRES project.....	2
2. The team.....	3
3. Requirements to use the program.....	3
4. When should I use OSIRES.....	4
5. Basic structure of the OSIRES program.....	5
5.1. Necessary steps and good practices.....	6
5.2. Features Already Implemented.....	7
5.2.1. Welcome screen.....	8
5.2.2. Files to compose the optimization.....	8
5.2.3. Values for optimization variables.....	9
5.2.4. Problem solution constraints.....	10

5.2.5. Optimization methods.....	11
5.2.6. Scatter and Scatter with Machine Learning.....	12
5.2.7. GRASP and GRASP with Machine Learning.....	12
5.2.8. Optimization Result.....	14
6. Example of optimization problem.....	14
7. Points for improvement identified.....	18
8. Change history and credit of shares.....	19
1. O que é o OSIRES.....	21
2. Quem somos.....	22
3. Requisitos para utilizar.....	22
4. Quando devo utilizar dessa ferramenta.....	23
5. Estrutura básica do programa OSIRES.....	24
5.1. Passo necessários e boas práticas.....	25
5.2. Recursos já Implementados.....	26
5.2.1. Tela de boas vindas.....	27
5.2.2. Arquivos para compor a otimização.....	27
5.2.3. Valores para as variáveis de otimização.....	28
5.2.4. Restrições da solução do problema.....	29
5.2.5. Métodos de otimização.....	30
5.2.6. Scatter e Scatter com Machine Learning.....	31
5.2.7. GRASP e GRASP com Machine Learning.....	31
5.2.8. Resultado da Otimização.....	33
6. Exemplo de problema de otimização.....	33
7. Pontos de melhoria identificados.....	37
8. Histórico de alterações e crédito das participações.....	38

1. The OSIRES project.

The OSIRES (Open Source Iterative Resolution for Event Simulation) project aims to be a “people-to-people” environment in the broadest form of that expression. People who want to work and share their experiences with process optimization. The “people” concept was designed for different demands, considering that a user with only knowledge of interacting with a graphical interface is able to use it, to more advanced users who have some programming knowledge, can change and implement their own optimization methods. Thinking about the term “for people”, it is considered that good graphical interface and programming practices in Python were used to make it easier for a creator of models, methods, techniques and optimization tools to test and disseminate to any audience they wish, without having to worry about software costs.

OSIRES contributes to assisting in the resolution of problems that involve the search within the solution space for simulation models for discrete events, without excluding the possibility of working with continuous behavioral variables, as this is within its scope of action. Preferably involving situations in which the number of all possible answers to a problem is not feasible to be analyzed with the amount of time and computational resources available.

Thinking about the current context for implementing the technologies necessary for industry 4.0 (I4.0), technical, computational and human resources require investment that can be prohibitive for a large number of companies. These reasons can range from a lack of knowledge on the subject, to a lack of resources to be invested in techniques that are not part of the routine of company managers.

Combining these two concepts: 1) techniques for improving the productivity of companies or teaching and research projects with the adoption of I4.0 elements, with 2) the need for lower costs for testing and adoption of disruptive monitoring, analysis and process optimization and/or production systems, there is a need to create a free code system for testing and adopting modeling, simulation and optimization methods.

In this context, the free access program JaamSim (Java Simulation, available at jaamsim.com) was selected in its free version (there is a paid PRO version) to be the computational modeling environment, and from this modeling, evaluate different scenarios to solve this problem to find a configuration that is equal to or close to an optimum point that meets the objective function and its respective restrictions.

This is a collaborative project, with the aim of disseminating the concepts of simulation and optimization, with sufficient maturity to be tested and modified by undergraduate and postgraduate students, as well as people directly related to production systems, who may be researchers or enthusiasts of area. Please feel free to use, distribute and change OSIRES. If possible, share your experiences so that we can continue to improve the project to meet expectations and demands, sending an email to wilson.trigueiro@ufsj.edu.br

2. The team

We are a group passionate about solving optimization problems related to production systems problems, which emerged in the Production Engineering course at the Federal University of São João del-Rei (Brazil), arising from a research project by Prof. Dr. Wilson Trigueiro de Sousa Júnior, who works with analysis, computational modeling and optimization of systems models, and the student researchers Alex de Andrade Soares and Luís Arthur de Assis Moraes, based at the Santo Antônio campus, Praça Frei Orlando n° 170, center, Department of Mechanical Engineering and Production, city of São João Del Rei, Minas Gerais, Brazil. This is one of the research projects within the CIMOS research group (Center for Innovation in Modeling and Systems Optimization, www.ufsj.edu.br/cimos), with information and updates about the project at: [https:// ufsj.edu.br/osires/](https://ufsj.edu.br/osires/) .

Even though there is a defined origin, the aim is to expand this project to other universities, companies or researchers/practitioners, creating an ecosystem for the creation, and dissemination of good practices and news in the context of digital tools for productive improvement, taking advantage of the precepts of Industry 4.0 and over. Therefore, we are open to forming partnerships with those interested in the subject.

3. Requirements to use the program

As technical requirements, we recommend prior knowledge of computational modeling for systems that involve the simulation of discrete events. For more information about this technique and methodology, it is suggested to read books and scientific articles in the area of Modeling and Discrete Event Simulation.

OSIRES was developed under Windows 11 environment, with Python 3.11 programming language, requiring the installation of this compatible version or higher. To install all the dependencies of the packages used in Python, in the program's root directory there is a notepad-type file "requirements.txt" with all

the requirements. To install them directly, the following command can be used, at the windows command prompt in administrator mode, generated within the program's root folder: "pip install -r requisites.txt". Tests in a Mac OS or Linux environment were not generated, even though they had a great chance of being compatible with these systems, due to the generalist nature with which it was programmed.

As it is a complement to the JaamSim program, it requires the installation of Java Development Kit 8 or higher. In tests with JDK 17, its use was satisfactory. The open source version of OpenJDK has not been tested, and there may also be compatibility.

Considering the average computational power of a user who has a personal computer, with processing by more than one core (multi-threading CPU), OSIRES is capable of being used for medium-sized problems on everyday machines, without the initial need investment in medium and large equipment.

4. When should I use OSIRES

OSIRES was designed to be used in problems involving the concept of complexity of the Non-Polynomial Hard type (in English, NP-Hard), in which there is neither time nor computational power sufficiently available for decision-making agents to be generated and evaluated. all possible combinations of the variables of the problem addressed.

With this type of problem in mind, optimization methods were implemented to generate and test good responses in an acceptable computational time, with the computational power of a personal computer. This way, the responses generated may be close to local optimum points, but there is no guarantee that they will be global optimum points (the best possible solution to the problem). Even without this guarantee, they are methods that help the decision-making agent in generating and evaluating an infinite number of combinations for the problems addressed.

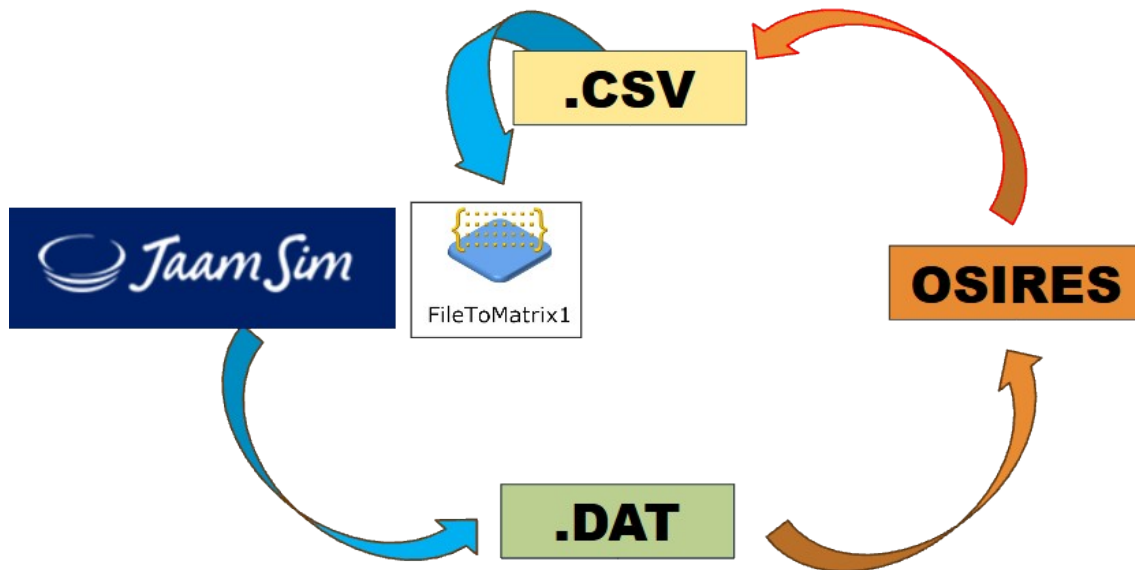
Between generating solutions and evaluating them to compare whether there is a better solution than the current solution (one optimization cycle), OSIRES works with optimization cycles of a minimum size of 100 solutions. Therefore, it is interesting to optimize problems whose solution space is greater than 100 elements. Otherwise, OSIRES can still be used, but it is also recommended to use isolated scenario generation methods. The number of 100 scenarios per cycle was determined as an empirical point, to reduce the number of recursive calls to the simulation model generated in JaamSim's java environment, considering that OSIRES is in Python, and this data transmission

between different Programming languages have a considerable computational cost.

5. Basic structure of the OSIRES program

OSIRES was designed to interact with simulation models generated in the JaamSim environment through the resource (“programming block”) present in the program called “File to matrix”. With this feature, JaamSim models are capable of generating results for different scenarios with just one configuration, from a “.CSV” file that contains the values for the desired variables. After reading the scenarios, the simulation results can be exported to a “.DAT” file. Figure 1 represents the simulation and optimization cycles.

Figure 1. Data structure and operation of OSIRES.



The OSIRES program utilized the following basic capabilities of JaamSim:

- 1) JaamSim, through the “File to Matrix” command block, is capable of generating the result of several configuration scenarios for the same simulation model with just one “run” or program call; and

- 2) JaamSim allows it to be called and executed by other programs in a mode that allows the absence of graphical output. In this way, it was used as a generator of simulations and results.

Thus, from the optimization modeling settings configured on the OSIRES screens, the selected optimization method generates values for the scenarios that are saved in the .CSV file, OSIRES calls the JaamSim execution in GUI-less mode and waits for the .DAT file is generated with the results of the simulation scenarios. With the .DAT values, OSIRES searches for the value of the best scenario and compares it with the best value previously stored, respecting the optimization criteria, whether or not there may be a better one in this optimization cycle. The process of generating values for the .CSV file is repeated if the stopping criterion, pressing the stop button, is not activated. If the stop button is pressed after generating the .CSV, the program waits for the JaamSim scenario generation cycle to finish to completely pause the program.

5.1. Necessary steps and good practices

In order to avoid problems during the simulation model optimization process, some “good practices” are suggested to prevent and evaluate possible errors before the optimization process.

For the discrete event simulation model, the following are suggested:

1) Generate the computational model and test it to verify that it has no errors and behaves as expected; and

2) When implementing the “File To Matrix” feature in JaamSim, carry out a test before using OSIRES to check how the model behaves in extreme situations for the optimization variables. This way, the optimization is prevented from generating failures due to a JaamSim simulation model with “bugs” and not due to the OSIRES optimization methods. Due to the inherent nature of OSIRES' interaction with JaamSim, OSIRES cannot identify whether there are problems in the modeling/simulation. We assume that the simulation is modeled correctly and has no problems for the variables and respective parameters defined for optimization. It is important to note that every model must be run at least once before being optimized to generate the .DAT file necessary for optimization.

3) Check how the simulation model verification and validation parameters are behaving for the system that is the focus of the study. A computational model with high variance of the output data may require measures to be taken, such as increasing the simulation time.

For optimization modeling, the following practices are suggested:

1) When testing more than one optimization method, especially hybrid ones that involve machine learning models, check that the .dat and .csv files do not contain too much “residue” from previous tests. By definition, the .dat file

stores the results of scenarios generated in previous rounds so that modeling agents/decision makers can evaluate how the data behaves over time and how the optimization results are behaving.

2) When re-using a file with the settings of an optimization already carried out, to be used in another optimization, on a new machine, pay attention to the fact that the path of the .csv, .dat and .cfg files are different, making it necessary to change the path of these files so that the previously configured optimization is reused for a new one. If this practice is not observed, OSIRES displays a red error screen when trying to proceed with the subsequent optimization configuration screens.

3) As the implemented optimization methods have functions that use randomness and do not generate, in most cases, the entire solution space and/or neighbors of a solution, it is possible that the optimal solution found is equal to an optimum point or that is very close to that point. In this way, it is up to the decision-making agent to evaluate the solution generated for the problem, observing the real possibilities in which the real system is inserted. Thus, the optimization response can be considered a “guideline” within the decision-making process, and not the final word on the given problem.

4) Every time before pressing the “Run” button, save the optimization model to avoid data loss and other issues.

These practices summarize, but do not prevent, other containment measures from being adopted to avoid unwanted results from the optimization process. For each level of programming, whether creating the simulation model or the optimization model, security levels must be created by users.

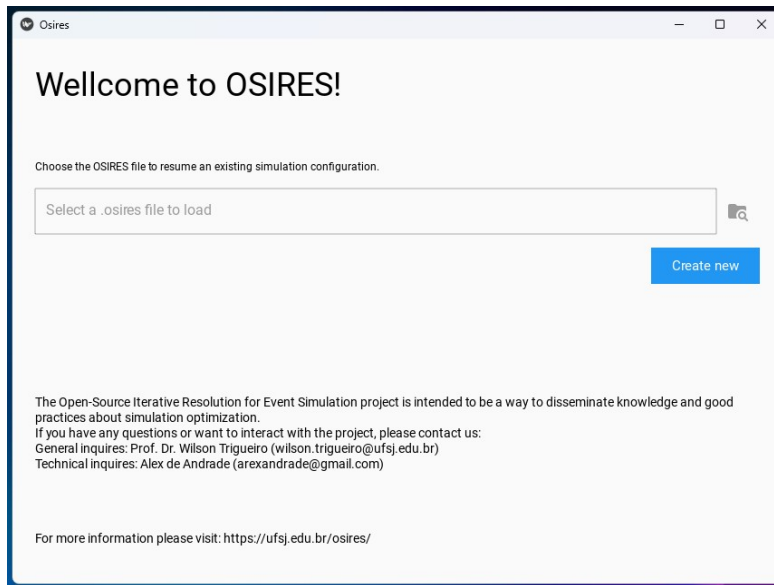
5.2. Features Already Implemented

OSIRES was designed with an easy-to-use graphical interface, so the user focuses on selecting the most important parameters to calibrate the optimization, without worrying about the computational resources necessary for creating scenarios, recursive calls and storing configurations and results.

5.2.1. Welcome screen

On the OSIRES home screen, you can create a new model or open a previously saved project. Basic information about the program is also displayed, allowing you to find supplementary material and contact the project managers.

Figure 2. OSIRES welcome screen, load or create project.

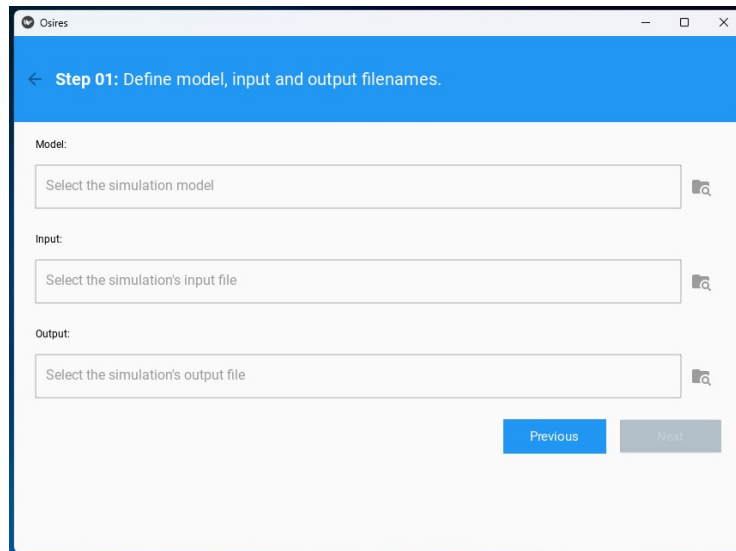


The screens for the GUI (Graphic User Interface) were generated from the module for Python Kivy and KivyMD. When the program is running, a prompt screen is automatically generated, where information is printed to know the status of some basic functions being executed and giving an interactive response to the user, to know what is being executed at that moment.

5.2.2. Files to compose the optimization

To perform the optimization, 3 files are required, namely: the simulation modeled in JaamSim, with the extension ".cfg"; the file containing possible scenarios to be evaluated by JaamSim, with the extension ".csv"; and the file containing the output values defined to be evaluated for each scenario (defined in the previous ".csv" file), this one in ".dat" format.

Figure 3. Files needed to optimize via OSIRES.



Osires

← Step 01: Define model, input and output filenames.

Model:

Select the simulation model

Input:

Select the simulation's input file

Output:

Select the simulation's output file

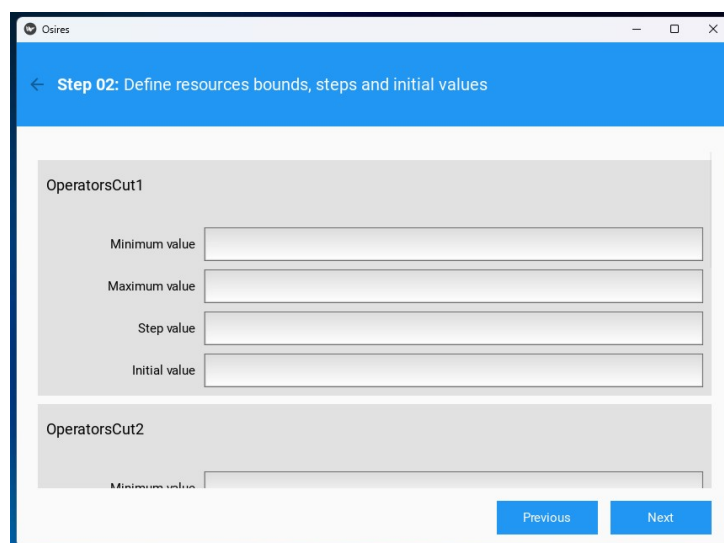
Previous Next

These are the 3 files that are generated for the initial testing of the modeling and scenario generation in JaamSim (best practices 1 and 2 described in section 5.1.). From the modeling and configuration for testing different scenarios in JaamSim, these 3 files are ready to be used for optimization via OSIRES.

5.2.3. Values for optimization variables

On the third screen, the parameters for each of the variables that are part of the problem solution are presented.

Figure 4. Configuration for optimization variables.



Osires

← Step 02: Define resources bounds, steps and initial values

OperatorsCut1

Minimum value

Maximum value

Step value

Initial value

OperatorsCut2

Minimum value

Previous Next

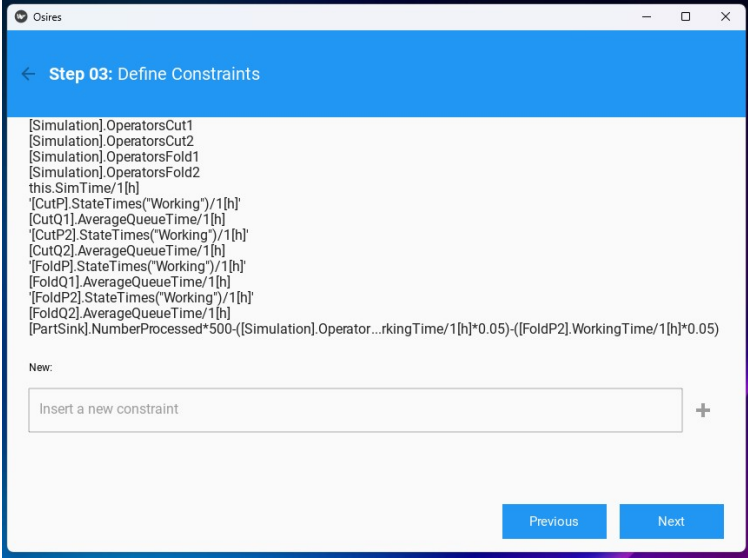
Variables are the resources available and necessary for processes to occur, so that, in order to be configured and optimized by OSIRES, they must be declared in the “.csv” file defined on the previous screen. The names and quantity of these variables are taken from the first line of this “.csv” file.

At this point, it is possible to define: the lowest possible value, the highest possible value, the amount of increment to be added or subtracted from the current value of the variable in order to consider that one response is different from another, and finally an initial value of the variable that may be close to an optimum already studied. This last parameter is interesting for some optimization methods that use an initial value to begin searches in the solution space, and are dependent on the initial value to arrive at a good response.

5.2.4. Problem solution constraints

In addition to the problem variables, it is possible to define other restrictions that do not involve resources, from expressions that use mathematical logic, involving the operators: '>', '<', '==', '+', '-', '*', '/', '!=', '**', '>=', '<=', '%', 'and', 'or', 'not' (greater than, less than, equal to, more than, less than, times, divided, except, raised to, greater than or equal to, less than or equal to, remainder of division, not equal to, logical and, logical or, and not).

Figure 5. Definition of optimization constraints.



The screenshot shows the OSIRES application window with the title bar 'Osires'. The main content area is titled 'Step 03: Define Constraints'. It contains a list of simulation variables: [Simulation].OperatorsCut1, [Simulation].OperatorsCut2, [Simulation].OperatorsFold1, [Simulation].OperatorsFold2, this.SimTime/1[h], '[CutP].StateTimes("Working")/1[h]', '[CutQ1].AverageQueueTime/1[h]', '[CutP2].StateTimes("Working")/1[h]', '[CutQ2].AverageQueueTime/1[h]', '[FoldP].StateTimes("Working")/1[h]', '[FoldQ1].AverageQueueTime/1[h]', '[FoldP2].StateTimes("Working")/1[h]', '[FoldQ2].AverageQueueTime/1[h]', and [PartSink].NumberProcessed*500-([Simulation].Operator...rkingTime/1[h]*0.05)-([FoldP2].WorkingTime/1[h]*0.05). Below the list is a section labeled 'New:' with a text input field containing the placeholder 'Insert a new constraint' and a plus icon. At the bottom right are 'Previous' and 'Next' buttons.

Unlike the variables that are taken as a response, which must be defined in the “.csv” file, the constraints can be any value from the simulation that is exported to the “.dat” file. This way, it is possible to include as a constraint any variable, whether dependent or independent, as long as it is in the simulation model. It is interesting to note that on this screen, sometimes the variables are

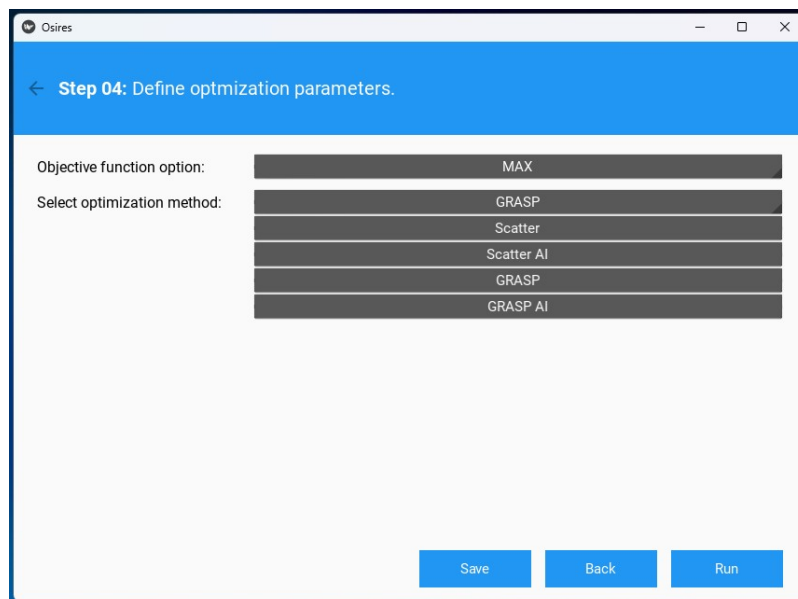
printed with the symbols (``) at the beginning and end of their presentation on this screen, however, to be used as part of a constraint equation or inequality, they must be written without these symbols.

To be possible to use a variable as a constraint, first it should be declared inside the JaamSim model at the Simulation>>Key Inputs>Run Output List to the printed at the .DAT file that can be read at the end of each simulation run.

5.2.5. Optimization methods

After defining possible restrictions, on the next screen it is possible to select a search method for optimization, as well as whether the objective function is to be maximized or minimized.

Figure 6. Definitions of optimization methods.



Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method:

- GRASP
- Scatter
- Scatter AI
- GRASP
- GRASP AI

Save Back Run

In the current version (ver. 2.0) two classes of optimization methods were implemented. The first refers to the classical optimization methods with scatter search and GRASP (Greedy Randomized Adaptive Search). They were chosen because they are simple (they do not require many adjustment parameters), with a chance of obtaining solutions close to or close to the global one and allow a population search, a desirable fact for the OSIRES structure because population-type searches facilitate the implementation of cycles with a size of 100 scenarios at a time.

5.2.6. Scatter and Scatter with Machine Learning

In the Scatter method (or scattered search) the values are generated randomly for the scenarios. Since by definition the same JaamSim model is loaded, the random number seed is the same for the simulation variables and processes, but the OSIRES random number seed changes over time, so the scenarios can change with different program calls, generating different scenarios.

Scatter was implemented with the following features:

1) As it is a “dispersed” method, it generates a good idea of how the problem’s solution space behaves. To take advantage of this feature, all generated and tested scenarios are stored in the .DAT file, and can be accessed with any text editor to read the data.

2) For version 2024-08 of JaamSim, it is possible to use the JaamSim CPU parallelism feature, enabling within the simulation model >>Simulation>>Multiple Runs>>Number of Threads, setting the number of Threads that the computer's CPU supports to equal or less.

5.2.7. GRASP and GRASP with Machine Learning

In the GRASP (Greedy Randomized Adaptive Search Procedure) method, an initial value is generated randomly, and from this initial solution, neighbors are generated. Since by definition the same JaamSim model is loaded, the random number seed is the same for the simulation variables and processes, but the OSIRES random number seed changes over time, so the scenarios can change with different program calls, generating different scenarios, also for this set of methods.

GRASP was implemented with the following features:

1) As it is a “population search” method, it generates a good idea of how the problem’s solution space behaves around an answer, which is why it takes advantage of the “initial suggestion” capacity in defining the optimization parameters.

2) For the 2024-08 version of JaamSim, it is possible to use the JaamSim CPU parallelism feature, enabling within the simulation model >>Simulation>>Multiple Runs>>Number of Threads, setting the number of Threads that the computer's CPU supports to be equal to or less, only for methods that use Machine Learning, with “pure” GRASP having to be executed in Single Thread.

3) For the correct configuration and execution of the GRASP + ML methods, the issues described in section 7, item 4 must be observed.

Figure 7. Machine-learning methods available for optimization.

Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: GRASP AI

Select AI method: RF

Minimum simulations for train AI: DT

Save Back Run

Figure 8. Parameter of AI optimization methods.

Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: Scatter AI

Select AI method: RF

Minimum simulations for train AI: Minimum 100 by 100

Save Back Run

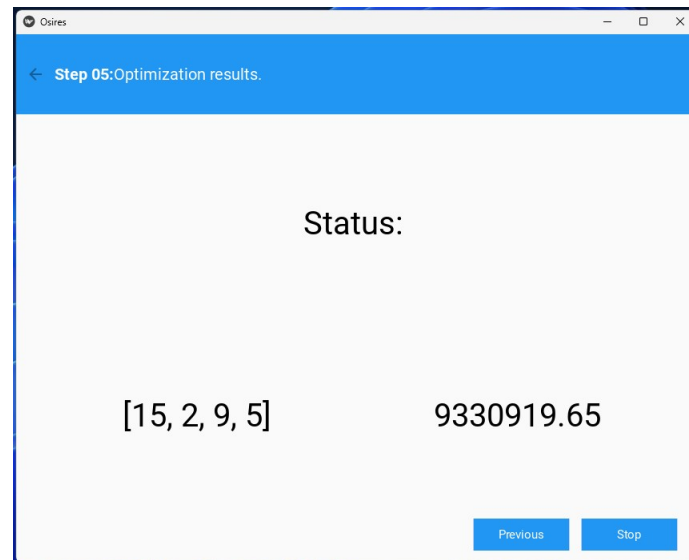
When selecting an AI method, a value must be entered as a parameter for the number of items to train the method. This value must be a multiple of 100, due to the cycle of generating values for the “.csv” file. If the model has already been trained previously, it is possible to enter the value “0” (zero), and the program understands that you want to use the model already trained and stored within the program.

On this same screen, for all methods, it is possible to save all the settings entered so far, by creating a file of the “.osires” type. By definition, this file can be read by any text editor and contains: the directory of the csv, dat and cfg files, the variable parameters, the restrictions, optimization settings and the best optimization response generated so far.

5.2.8. Optimization Result

On the final screen of the program, the values considered optimal up to this point are presented.

Figure 8. Results of the optimization method.



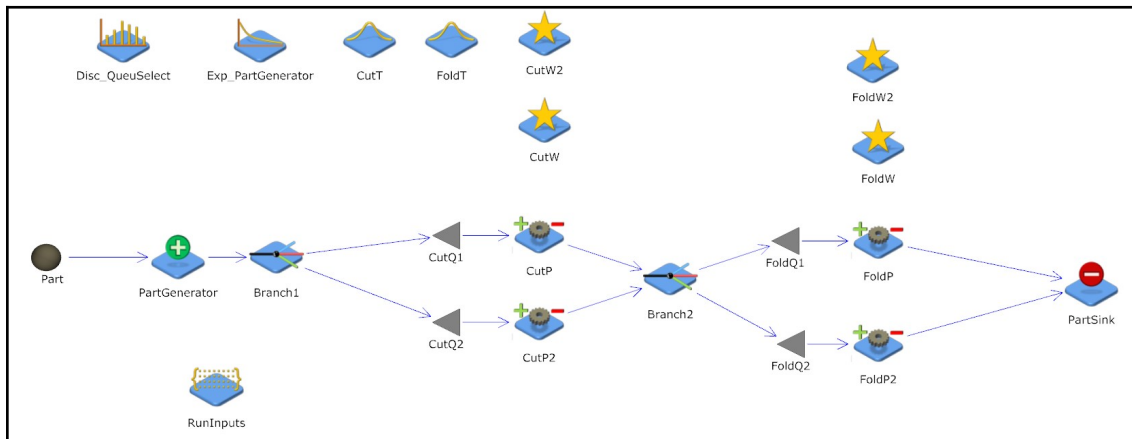
This shows the values of the response variables and the value of the objective function for this configuration. This value is the best value found so far and, by definition, respects all previously defined restrictions. Every time this result is updated with a new optimum point, this result updates the values stored in the ".osires" file.

6. Example of optimization problem

The parts arrive at a production line with an exponential interarrival time, with a mean of 8 min and a maximum value of 10 minutes, and were modified by the cutting and bending processes that last a normal distribution, with a mean of 10 min and a standard deviation of 1 minute each. The first process has two stations, as does the second in parallel. As the parts pass through the process, they are able to go to each process queue randomly with the same probability. It is assumed that there is a demand for 90,000 parts to be processed each year. Each part produced generates a profit of \$500, and the costs are: \$20,000 for each operator, \$0.05 times: the total time consumed,

waiting time and total workstation time. Determine the number of operators that maximize the net profit. The screen representation of this model in JaamSim is shown. This model is available at www.ufsj.edu.br/osires/.

Figure 9. Modeling the production line problem in JaamSim.



Considering that the discrete event model of the production line problem has been tested and the csv, dat and cfg files are correctly configured for integration with OSIRES, we move on to configuring the optimization model.

For better detailing and understanding of this problem and modeling, there are videos in English:

- 1)
- 2)

Figure 10. Defining the paths for the csv, dat and cfg files.

After selecting the files, OSIRES reads the first line of the .csv file and determines the number of variables that will be optimized. In this case, there are four operators: type 1 cut, type 2 cut, type 1 fold and type 2 fold.

Figure 11. Definition of optimization variable parameters.

Osires

← Step 02: Define resources bounds, steps and initial values

OperatorsCut1

Minimum value 1

Maximum value 50

Step value 1

Initial value 1

OperatorsCut2

Minimum value 1

Previous Next

The next step is to define, if necessary, the acceptance constraints for the optimization responses, the problem constraints. In this case, since the lower bound for each variable is equal to one, the first constraint is already met, with the minimum value of the sum of the four variables being equal to four, with none equal to zero.

Figure 12. Definition of variable constraint parameters.

Osires

← Step 03: Define Constraints

[Simulation].OperatorsCut1
[Simulation].OperatorsCut2
[Simulation].OperatorsFold1
[Simulation].OperatorsFold2
this.SimTime/1[h]
[CutP].StateTimes("Working")/1[h]
[CutQ1].AverageQueueTime/1[h]
[CutP2].StateTimes("Working")/1[h]
[CutQ2].AverageQueueTime/1[h]
[FoldP].StateTimes("Working")/1[h]
[FoldQ1].AverageQueueTime/1[h]
[FoldP2].StateTimes("Working")/1[h]
[FoldQ2].AverageQueueTime/1[h]
[PartSink].NumberProcessed*500-([Simulation].Operator...rkingTime/1[h]*0.05)-([FoldP2].WorkingTime/1[h]*0.05)

New:

Insert a new constraint +

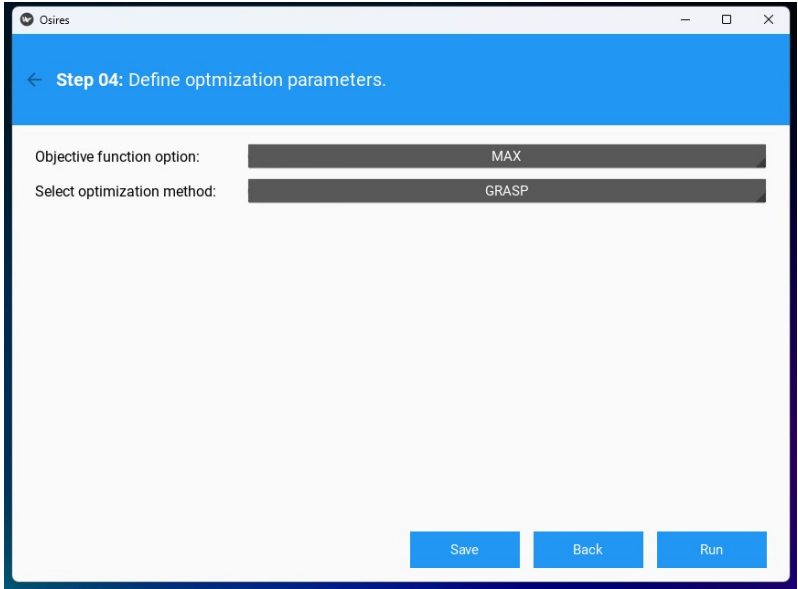
#0 [Simulation].OperatorsCut1 + [Simulation].OperatorsCut2 <= 50 -

#1 [Simulation].OperatorsFold1 + [Simulation].OperatorsFold2 <= 50 -

Previous Next

For educational purposes, two restrictions were represented, where the variables must have their sums less than 50. For the present problem, it was defined that the maximum physical space capacity for the production line would be 100 employees.

Figure 13: Optimization method selection



Osires

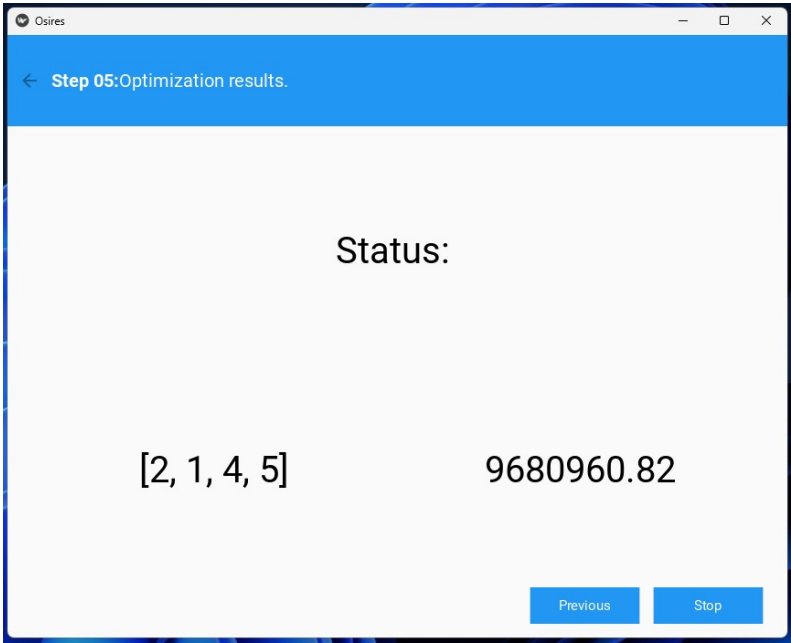
← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: GRASP

Save Back Run

Figure 14: Optimization running and status



Osires

← Step 05: Optimization results.

Status:

[2, 1, 4, 5] 9680960.82

Previous Stop

7. Points for improvement identified

At the time of its release, for version 3.0, the following points were identified to be observed for the next version:

- 1) When using the minimization function, two numbers may be generated for the last optimization variable. As an initial containment measure, a restriction can be created to prevent these values from exceeding the maximum limit of the optimization variable. This factor was very rarely identified in the models tested.
- 2) When entering data, OSIRES only accepts changes to data before the “Run” button is pressed. After pressing this button, all parameters are set as originally configured on the previous screens.

8. Change history and credit of shares

OSIRES (ver.01)	
Activities	Responsible
Project design, management and maintenance	Wilson Trigueiro de S. Jr.
Definition of “front end”	Alex de Andrade Soares
Definition of Reload screens, files, parameters, restrictions and optimization	Alex de Andrade Soares
Integration of Python with JaamSim 2022-06, when performing simulations.	Alex de Andrade Soares
Implementation of the “Scatter Search” method	Alex de Andrade Soares
Implementation of the “Scatter Search + ML” method	Alex de Andrade Soares
Implementation of the “GRASP” method	Alex de Andrade Soares
Implementation of the “GRASP + ML” method	Alex de Andrade Soares
Implementation of method for navigation	Luís Arthur de Assis Moraes

between fields	
Example presentation in the restriction field	Luís Arthur de Assis Moraes
OSIRES (ver.02)	
Addition of dynamic graphical method, with the output of the optimization result, per iteration.	Luís Arthur de Assis Moraes
Correction of errors in handling input data from simulation files.	Luís Arthur de Assis Moraes
Update the version of JaamSim used to 2024-08.	Luís Arthur de Assis Moraes

1. O que é o OSIRES

O projeto OSIRES (Open Source Iterative Resolution for Event Simulation) tem o intuito de ser um ambiente “de pessoas para pessoas” na forma mais ampla dessa expressão. Pessoas que querem trabalhar e compartilhar suas experiências com otimização de processos. O conceito “de pessoas” foi pensado para diferentes demandas, considerando que um usuário com apenas conhecimento de interação com interface gráfica seja capaz de utilizar, até usuários mais avançados que possuem algum conhecimento de programação, possam alterar e implementar seus próprios métodos de otimização. Pensando no termo “para pessoas”, é considerado que foram utilizados boas práticas de interface gráfica e de programação em Python para facilitar que um criador de modelos, métodos, técnicas e ferramentas de otimização, possam testar e disseminar para quaisquer públicos que desejar, sem ter que preocupar com os custos de software.

O OSIRES vem contribuir para o auxílio na resolução de problemas que envolvam a busca dentro do espaço solução de modelos de simulação a eventos discretos, não excluindo a possibilidade de trabalhar com variáveis de comportamento contínuos, pois está dentro de seu escopo de atuação. Preferencialmente de envolver situações em que a quantidade de todas as possíveis respostas para um problema não seja viável de ser analisada com a quantidade de tempo e recursos computacionais disponíveis.

Pensando no atual contexto para a implementação das tecnologias necessárias para a indústria 4.0 (I4.0), recursos de ordem técnica, computacional e humana demandam um aporte de investimento que pode ser proibitivo para uma grande quantidade de empresas. Estes motivos podem variar desde a falta de conhecimento sobre o tema, até a escassez de recursos para serem investidos em técnicas que não fazem parte da rotina dos gestores das empresas.

Juntando esses dois conceitos: 1) técnicas para melhoria da produtividade das empresas ou projetos de ensino e pesquisa com a adoção de elementos da I4.0, com 2) a necessidade de menor custo para teste e adoção de técnicas disruptivas de acompanhamento, análise e otimização de processo e/ou sistemas produtivos, surge a necessidade de criação de um sistema de código livre para teste e adoção de métodos de modelagem, simulação e otimização.

Nesse contexto, foi selecionado o programa de acesso livre JaamSim (Java Simulation, disponível em jaamsim.com), em sua versão gratuita (existe uma versão PRO paga) para ser o ambiente de modelagem computacional, e a partir dessa modelagem, avaliar diferentes cenários para a resolução desse

problema para encontrar uma configuração que seja igual ou próxima de um ponto de ótimo que atenda à função objetivo e suas respectivas restrições.

Este é um projeto colaborativo, com o intuito de difundir os conceitos de simulação e otimização, com maturidade suficiente para ser testado e modificado por estudantes de graduação e pós-graduação, assim como pessoas diretamente relacionadas com sistemas produtivos, podem ser pesquisadores ou entusiasta da área. Sinta-se à vontade para utilizar, distribuir e alterar o OSIRES. Se possível compartilhe suas experiências para continuarmos a melhorar o projeto para atender as expectativas e demandas, enviando um e-mail para wilson.trigueiro@ufsj.edu.br

2. Quem somos

Somos um grupo apaixonado por resolver problemas de otimização relacionados a sistemas produtivos, que surgiu no curso de Engenharia de Produção da Universidade Federal de São João del-Rei, oriundo de projeto de pesquisa do Prof. Dr. Wilson Trigueiro de Sousa Júnior, que trabalha com análise, modelagem computacional e otimização de modelos de sistemas, e os alunos pesquisadores Alex de Andrade Soares e Luís Arthur de Assis Moraes, lotados no campus Santo Antônio, praça Frei Orlando nº 170, centro, Departamento de Engenharia Mecânica e Produção, cidade de São João Del Rei, Minas Gerais, Brasil. Este é um dos projetos de pesquisa que está dentro do grupo de pesquisa CIMOS (Centro de Inovação em Modelagem e Otimização de Sistemas, www.ufsj.edu.br/cimos), com informações e atualizações sobre o projeto em: <https://ufsj.edu.br/osires/>.

Por mais que exista uma origem definida, é pretendida a expansão desse projeto para outras universidades, empresas ou pessoas, criando um ecossistema de criação, disseminação e divulgação de boas práticas e novidades no contexto de ferramentas digitais para a melhoria produtiva, aproveitando os preceitos da I4.0. Dessa forma estamos abertos a fazer parcerias com interessados no assunto.

3. Requisitos para utilizar

Como requisitos técnicos, recomendamos o conhecimento prévio sobre modelagem computacional para sistemas que envolvam a simulação de eventos discretos. Para mais informações sobre essa técnica e metodologia,

fica como sugestão a leitura de livros e artigos científicos da área de Modelagem e Simulação de Eventos Discretos.

O OSIRES foi desenvolvido em ambiente Windows 11, com linguagem de programação Python 3.11, necessitando da instalação dessa versão ou superior compatível. Para a instalação de todas as dependências dos pacotes utilizados em Python, no diretório raiz do programa existe um arquivo do tipo bloco de notas “requirements.txt” com todos os requisitos. Para a instalação de forma direta dos mesmos, pode ser usado o seguinte comando, no prompt de comando em modo administrador, gerado dentro da pasta raiz do programa: “pip install -r requirements.txt”. Testes em ambiente Mac OS ou Linux não foram gerados, mesmo havendo uma grande chance de serem compatíveis com esses sistemas, devido a natureza generalista com que foi programado.

Por ser um complemento para o programa JaamSim, exige a instalação do Java Development Kit 8 ou superior. Em testes com o JDK 17, a sua utilização foi satisfatória. Não foi testado a versão de código livre OpenJDK, podendo haver também compatibilidade.

Considerando o poder computacional médio de um usuário que possui um computador pessoal, com processamento por mais de um núcleo (multi-threading CPU), o OSIRES é capaz de ser utilizado para problemas de médio porte em máquinas do uso diário, sem a necessidade inicial de investimento em equipamentos de médio e grande porte.

4. Quando devo utilizar dessa ferramenta

O OSIRES foi idealizado para ser utilizado em problemas que envolvam o conceito de complexidade do tipo Não-Polinomial Difíceis (em inglês NP-Hard), em que não existe nem tempo ou poder computacional suficientemente disponíveis pelos agentes tomadores de decisão para serem geradas e avaliadas todas as possíveis combinações das variáveis do problema tratado.

Com este tipo de problema em mente, foram implementados métodos de otimização para se gerar e testar boas respostas em um tempo computacional aceitável, com o poder computacional de um computador pessoal. Dessa forma, as respostas geradas podem estar próximas de pontos de ótimos locais, mas não há a garantia de serem pontos de ótimo globais (a melhor solução possível para o problema). Mesmo sem essa garantia, são métodos que auxiliam o agente tomador de decisão na geração e avaliação de uma infinidade de combinações para os problemas tratados.

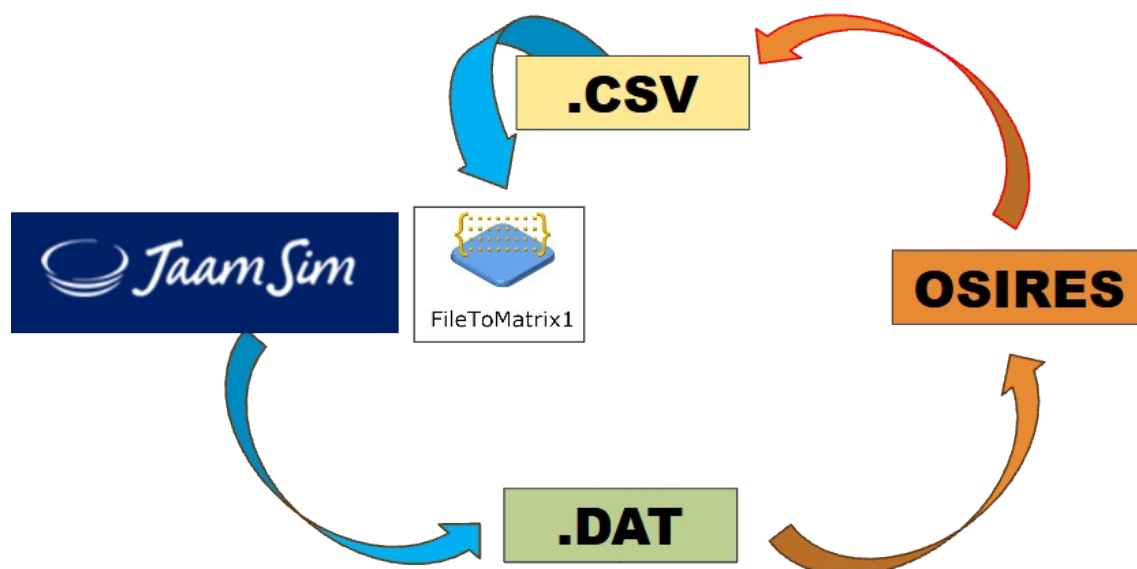
Entre a geração de soluções e a respectiva avaliação para comparar se existe uma melhor que a solução corrente (um ciclo da otimização), o OSIRES trabalha com ciclos de otimização de tamanho mínimo de 100 soluções. Dessa

forma é interessante otimizar problemas cujo espaço solução seja maior que 100 elementos. Caso contrário, o OSIRES ainda sim pode ser utilizado, mas é recomendado também a utilização de métodos de geração de cenários isolados. A quantidade de 100 cenários por ciclo foi determinada como um ponto empírico, para reduzir a quantidade de chamadas recursivas para o modelo de simulação gerado em ambiente java do JaamSim, tendo em vista que o OSIRES é em Python, e essa transmissão de dados entre diferentes linguagens de programação tem um custo computacional considerável.

5. Estrutura básica do programa OSIRES

O OSIRES foi projetado para interagir com os modelos de simulação gerados no ambiente JaamSim através do recurso (“bloco de programação”) presente no programa denominado “File to matrix”. Com este recurso, os modelos do JaamSim são capazes de gerar os resultados para diferentes cenários com apenas uma configuração, a partir de um arquivo “.CSV” que contenha os valores para as variáveis desejadas. Após ler os cenários, os resultados das simulações podem ser exportados para um arquivo do tipo “.DAT”. A Figura 1 representa os ciclos de simulação e de otimização.

Figura 1. Estrutura de dados e funcionamento do OSIRES.



O programa de OSIRES utilizou das seguintes capacidades básicas do JaamSim:

- 1) O JaamSim através do bloco de comando “File to Matrix” é capaz de gerar o resultado de vários cenários de configuração para um mesmo modelo de simulação com apenas uma “corrida” ou chamada do programa; e
- 2) O JaamSim permite que ele seja chamado e executado por outros programas em modo que permite a ausência de saída gráfica. Dessa forma ele foi utilizado como gerador de simulações e resultados.

Assim, a partir das configurações da modelagem de otimização configuradas nas telas do OSIRES, o método de otimização selecionado gera valores para os cenários que são salvos no arquivo .CSV, o OSIRES chama a execução do JaamSim no modo sem interface gráfica e espera que o arquivo .DAT seja gerado com o resultado dos cenários das simulações. Com os valores do .DAT, o OSIRES busca o valor do melhor cenário e compara com o melhor valor até então armazenado, respeitando os critérios de otimização, podendo ou não existir um melhor nesse ciclo de otimização. O processo de geração dos valores para o arquivo .CSV se repete se o critério de parada, de apertar o botão de parada, não for acionado. Se o botão de parada for acionado depois da geração do .CSV, o programa espera que o ciclo de geração de cenários do JaamSim seja terminado para a pausa total do programa.

5.1. Passo necessários e boas práticas

Com o intuito de se evitar problemas durante o processo de otimização dos modelos de simulação, são sugeridas algumas “boas práticas” para se prevenir e avaliar possíveis erros antes do processo de otimização.

Para o modelo de simulação de eventos discretos, são sugeridos:

- 1) Gerar o modelo computacional e testar para verificar se o mesmo não possui erros e se comporta como esperado; e
- 2) Ao se implementar o recurso “File To Matrix” no JaamSim, fazer um teste antes de usar o OSIRES para se verificar como o modelo se comporta em situações de extremo para as variáveis da otimização. Dessa forma se evita que a otimização gere falhas por conta de um modelo de simulação do JaamSim com “bugs” e não por conta dos métodos de otimização do OSIRES. Pela natureza inerente de interação do OSIRES com o JaamSim, o OSIRES não consegue identificar se existem problemas na modelagem/simulação. Partimos do pressuposto que a simulação está modelada da maneira correta e não possui problemas para as variáveis e os respectivos parâmetros definidos para otimização. É importante ressaltar que todo modelo deve ser rodado pelo menos uma vez antes de ser otimizado para se gerar o arquivo .DAT necessário para a otimização.
- 3) Verificar como os parâmetros de verificação e validação do modelo de simulação estão se comportando para o sistema foco do estudo. Um modelo

computacional com alta variância dos dados de saída pode exigir que medidas sejam tomadas, a exemplo aumentar o tempo da simulação.

Para a modelagem da otimização, são sugeridas as seguintes práticas:

- 1) Ao se testar mais de um método de otimização, em especial os híbridos que envolvem modelos de aprendizagem de máquina, verificar se os arquivos .dat e .csv não possuem muito “resíduo” dos testes realizados anteriormente. Por definição, o arquivo .dat armazena os resultados dos cenários gerados em rodadas anteriores para que os agentes modeladores/tomadores de decisão possam avaliar como os dados se comportam no tempo e como os resultados da otimização estão se comportando.
- 2) Ao se re-utilizar um arquivo com as configurações de uma otimização já realizada, para ser aproveitada em uma outra otimização, em uma nova máquina, se atentar para o fato de que o caminho dos arquivos .csv, .dat e .cfg são diferentes, sendo necessário a mudança do caminho desses arquivos para que a otimização previamente configurada seja reutilizada para uma nova. Caso essa prática não seja observada, o OSIRES dá uma tela vermelha de erro, ao se tentar prosseguir com as telas seguintes da configuração da otimização.
- 3) Como os métodos de otimização implementados possuem funções que utilizam de aleatoriedade e não geram, na maioria das vezes, todo o espaço solução e/ou vizinhos de uma solução, é possível que a solução ótima encontrada seja igual a um ponto de ótimo ou que esteja bem próximo de tal ponto. Dessa maneira, cabe ao agente tomador de decisão a avaliação da solução gerada para o problema, observando as reais possibilidades as quais o sistema real está inserido. Assim a resposta da otimização pode ser considerado um “norte” dentro do processo de tomada de decisão, e não a palavra final sobre o dado problema.
- 4) Toda vez antes de apertar o botão “Run”, salvar o modelo de otimização, para evitar perdas de dados.

Essas práticas resumem, mas não impedem que outras medidas de contenção sejam adotadas para se evitar resultados não desejados pelo processo de otimização. Para cada nível de programação, seja a criação do modelo de simulação ou do modelo de otimização, níveis de segurança devem ser criados pelos usuários.

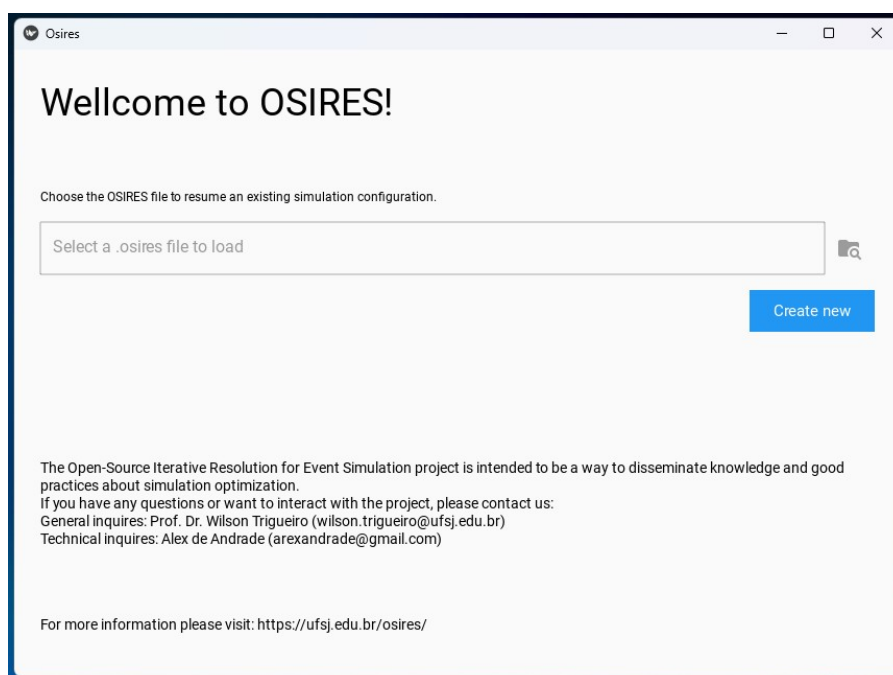
5.2. Recursos já Implementados

O OSIRES foi projetado com Interface gráfica de fácil uso, dessa forma o usuário foca em selecionar os parâmetros mais importantes para calibrar a otimização, sem se preocupar com os recursos computacionais necessários para a criação de cenários, chamadas recursivas e armazenamento de configurações e resultados.

5.2.1. Tela de boas vindas.

Na tela inicial do OSIRES é possível a criação de um modelo novo ou da abertura de um projeto já salvo. Também são apresentadas informações básicas sobre o programa para se encontrar material suplementar e contato com os gestores do projeto.

Figura 2. Tela de boas vindas do OSIRES, carregar ou criar projeto.



As telas para o GUI (Graphic User Interface) foram geradas a partir do módulo para Python Kivy e KivyMD. Quando o programa está sendo executado, uma tela de prompt é gerada automaticamente, onde informações são impressas para se saber o status de algumas funções básicas sendo executadas e dando uma resposta interativa para o usuário, para saber o que está sendo executado naquele momento.

5.2.2. Arquivos para compor a otimização

Para realizar a otimização, 3 arquivos são necessários, a saber: a simulação modelada no JaamSim, com extensão “.cfg”; o arquivo contendo possíveis cenários a serem avaliados pelo JaamSim, com extensão “.csv”; e o arquivo que contém os valores de saída definidos para serem avaliados para cada cenário (definido no arquivo anterior “.csv”), este com formato “.dat”.

Figura 3. Arquivos necessários para otimizar via OSIRES.

The screenshot shows the 'Step 01: Define model, input and output filenames.' window in the OSIRES application. It features three input fields with file selection icons: 'Model' (labeled 'Select the simulation model'), 'Input' (labeled 'Select the simulation's input file'), and 'Output' (labeled 'Select the simulation's output file'). At the bottom right, there are 'Previous' and 'Next' buttons.

Estes são os 3 arquivos que são gerados para o teste inicial da modelagem e teste de cenários no JaamSim (boas práticas 1 e 2 descritas na seção 5.1.). A partir da modelagem e configuração para teste de diferentes cenários no JaamSim, estes 3 arquivos estão prontos para serem utilizados para a otimização via OSIRES.

5.2.3. Valores para as variáveis de otimização

Na terceira tela, são apresentados os parâmetros para cada uma das variáveis que fazem parte da solução do problema.

Figura 4. Configuração para as variáveis de otimização.

The screenshot shows the 'Step 02: Define resources bounds, steps and initial values' window in the OSIRES application. It displays two sections for resource configuration: 'OperatorsCut1' and 'OperatorsCut2'. Each section contains four input fields: 'Minimum value', 'Maximum value', 'Step value', and 'Initial value'. At the bottom right, there are 'Previous' and 'Next' buttons.

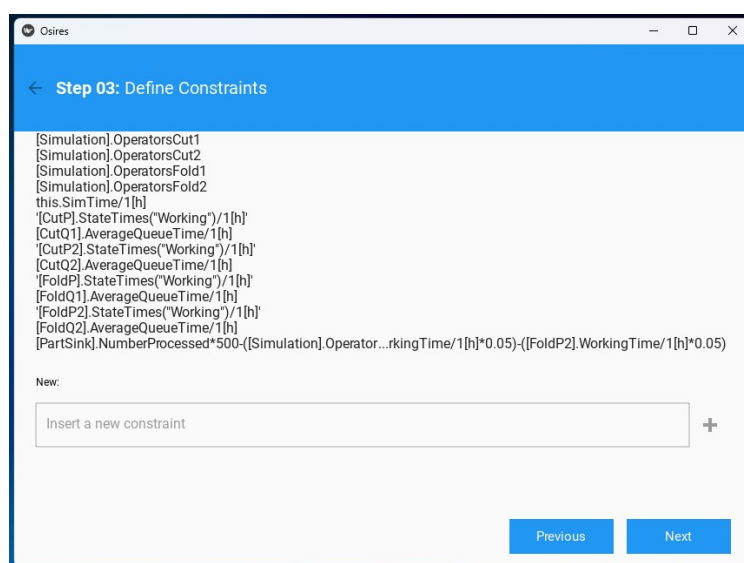
As variáveis são os recursos disponíveis e necessários para que os processos aconteçam, de forma que os mesmos, para serem configurados e otimizados pelo OSIRES, devem ser declarados obrigatoriamente no arquivo “.csv” definido na tela anterior. Os nomes e a quantidade dessas variáveis são retirados da primeira linha desse arquivo “.csv”.

Nesse momento é possível definir: o menor valor possível, o maior valor possível, a quantidade de incremento a ser acrescentado ou diminuído do valor corrente da variável de forma a considerar que é uma resposta é diferente da outra e por último um valor inicial da variável que pode ser próximo de um ótimo já estudado. Este último parâmetro é interessante para alguns métodos de otimização que utilizam de um valor inicial para começar as buscas no espaço solução, e são dependentes do valor inicial para se chegar em uma boa resposta.

5.2.4. Restrições da solução do problema

Além das variáveis do problema, é possível definir outras restrições que não envolvam os recursos, a partir de expressões que se utilizam de lógica matemática, envolvendo os operadores: '>', '<', '==', '+', '-', '*', '/', '!=', '**', '>=', '<=', '%', 'and', 'or', 'not' (maior, menor, igual, mais, menos, vezes, dividido, exceto, elevado, maior ou igual, menor ou igual, resto da divisão, não igual, lógica e, lógica ou, e não).

Figura 5. Definição das restrições de otimização.



Ao contrário das variáveis que são tomadas como resposta, que devem estar definidas no arquivo “.csv”, as restrições podem ser qualquer valor da

simulação que forem exportados para o arquivo “.dat”. Dessa forma é possível incluir como restrição qualquer variável, seja ela dependente ou independente, mas que esteja no modelo da simulação. É interessante comentar que nesta tela, as vezes as variáveis são impressas com os símbolos (“”) no início e fim de sua apresentação nesta tela, porém para serem utilizadas como parte de uma equação ou inequação de restrição, elas devem ser escritas sem este símbolo.

Para que seja possível usar uma variável como restrição, primeiro ela deve ser declarada dentro do modelo JaamSim em “Simulation>>Key Inputs>Run Output List” para ser impressa no arquivo .DAT que pode ser lido no final de cada execução da simulação.

5.2.5. Métodos de otimização

Após a definição de possíveis restrições, na tela seguinte é possível selecionar um método de busca para otimização, assim como se é pretendido maximizar ou minimizar a função objetivo.

Figura 6. Definições dos métodos de otimização.

Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method:

- GRASP
- Scatter
- Scatter AI
- GRASP
- GRASP AI

Save Back Run

Na versão atual (ver. 2.0) foram implementadas duas classes de métodos de otimização. A primeira, referente aos métodos clássicos de otimização com a busca dispersa (scatter search) e GRASP (Greedy Randomized Adaptive Search). Elas foram escolhidas por serem simples (não necessitam de muitos parâmetros de ajustes), com chance de obtenção de soluções próximas ou perto do global e permitirem uma busca populacional, fato este desejável para a estrutura do OSIRES porque devido à buscas do tipo

populacional facilitarem a implementação de ciclos com tamanho de 100 cenários por vez.

5.2.6. Scatter e Scatter com Machine Learning

No método Scatter (ou busca dispersa) os valores são gerados de forma aleatória para os cenários. Como por definição um mesmo modelo do JaamSim é carregado, a semente dos números aleatórios é a mesma para as variáveis e processos da simulação, mas a semente dos números aleatórios do OSIRES muda com o tempo, assim os cenários podem mudar com diferentes chamadas do programa, gerando diferentes cenários.

O Scatter foi implementado com as seguintes características:

1) Por ser um método “disperso” gera uma boa ideia de como o espaço solução do problema se comporta. Para aproveitar essa característica, todos os cenários gerados e testados são armazenados no arquivo .DAT, e podem ser acessados com qualquer editor de texto para a leitura dos dados.

2) Para a versão 2024-08 do JaamSim, é possível utilizar o recurso de paralelismo via CPU do JaamSim, habilitando dentro do modelo de simulação >>Simulation>>Multiple Runs>>Number of Threads colocando igual ou menor o número de Threads que a CPU do computador suportar.

5.2.7. GRASP e GRASP com Machine Learning

No método GRASP (Greedy Randomized Adaptive Search Procedure) um valor inicial é gerado de forma aleatória, e a partir dessa solução inicial, vizinhos são gerados. Como por definição um mesmo modelo do JaamSim é carregado, a semente dos números aleatórios é a mesma para as variáveis e processos da simulação, mas a semente dos números aleatórios do OSIRES muda com o tempo, assim os cenários podem mudar com diferentes chamadas do programa, gerando diferentes cenários, também para este conjunto de métodos.

O GRASP foi implementado com as seguintes características:

1) Por ser um método “de busca populacional” gera uma boa ideia de como o espaço solução do problema se comporta ao entorno de uma resposta, por isso ele se aproveita da capacidade de “sugestão inicial” na definição dos parâmetros de otimização.

2) Para a versão 2024-08 do JaamSim, é possível utilizar o recurso de paralelismo via CPU do JaamSim, habilitando dentro do modelo de simulação >>Simulation>>Multiple Runs>>Number of Threads colocando igual ou menor o número de Threads que a CPU do computador suportar, apenas para os métodos que se utilizam de Machine Learning, sendo que o GRASP “puro” tem que ser executado em Single Thread.

3) Para a correta configuração e execução dos métodos GRASP + ML, devem ser observadas as questões descritas na seção 7, item 4.

Figura 7. Métodos de machine-learning disponíveis para a otimização.

Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: GRASP AI

Select AI method: RF, DT, KN, GP

Minimum simulations for train AI: GP

Save Back Run

Figura 8. Parâmetro dos métodos de otimização com IA.

Osires

← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: Scatter AI

Select AI method: RF

Minimum simulations for train AI: Minimum 100 by 100

Save Back Run

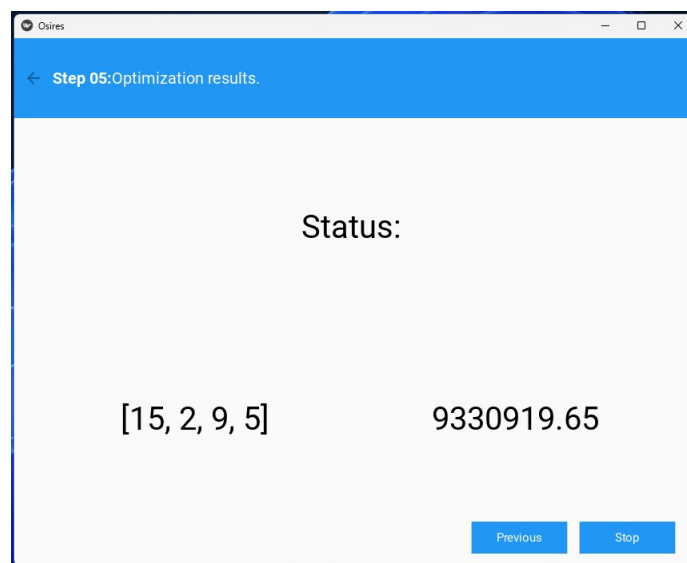
Quando selecionado um método com IA, um valor deve ser informado como parâmetro para a quantidade de itens para treinamento do método. Este valor deve ser um múltiplo de 100, devido ao ciclo de geração de valores para o arquivo “.csv”. Caso o modelo já tenha sido treinado anteriormente, é possível digitar o valor “0” (zero), e o programa entende que se deseja utilizar do modelo já treinado e armazenado dentro do programa.

Nesta mesma tela, para todos os métodos, é possível salvar todas as configurações inseridas até o momento, com a criação de um arquivo do tipo “.osires”. Por definição esse arquivo pode ser lido por qualquer editor de texto e contém: o diretório dos arquivos csv, dat e cfg, os parâmetros das variáveis, as restrições, configurações de otimização e a melhor resposta de otimização gerada até o momento.

5.2.8. Resultado da Otimização

Na tela final do programa, são apresentados os valores considerados como ótimos até o presente momento.

Figura 8. Resultados do método de otimização.



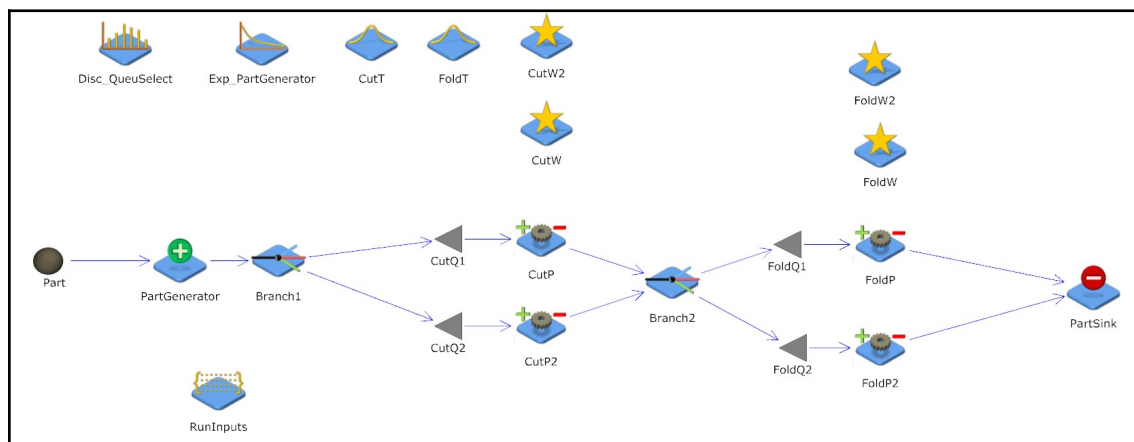
Dessa forma são apresentados os valores das variáveis resposta e o valor da função objetivo para esta configuração. Este valor é o melhor valor encontrado até o momento e, por definição, respeita todas as restrições definidas anteriormente. Toda vez que esse resultado é atualizado com um novo ponto de ótimo, este resultado atualiza os valores armazenados no arquivo “.osires”.

6. Exemplo de problema de otimização

As peças chegam em uma linha de produção com tempo entre chegadas exponencial, com média de 8 min e valor máximo de 10 minutos e foram modificadas pelos processos de corte e dobra que duram uma distribuição normal, com média de 10 min e desvio padrão de 1 minuto cada. O primeiro processo possui duas estações, assim como o segundo em paralelo. À

medida que as peças passam pelo processo, consegue-se ir para cada fila de processo de forma aleatória com a mesma probabilidade. Presume-se que exista uma demanda de 90.000 peças para serem processadas a cada ano. Cada peça produzida gera um lucro de \$500, e os custos são: \$20.000 para cada operador, \$0,05 vezes: o tempo total consumido, tempo de espera e tempo total da estação de trabalho. Determine o número de operadores que maximizam o lucro líquido. É apresentado a tela da representação desse modelo no JaamSim. Este modelo está disponível no site www.ufsj.edu.br/osires/.

Figura 9. Modelagem do problema da linha de produção no JaamSim.



Considerando que o modelo de eventos discretos do problema da linha de produção foi testado e os arquivos csv, dat e cfg estão corretamente configurados para integração com o OSIRES, passamos para a configuração do modelo de otimização.

Para melhor detalhamento e entendimento desse problema e modelagem, existem os vídeos em português:

- 1) <https://youtu.be/-WiyQH1FnAI>
- 2) <https://youtu.be/AoAaJmKnVYw>

Figura 10. Definição dos caminhos para os arquivos csv, dat e cfg.

The screenshot shows the 'Step 01: Define model, input and output filenames' window in the Osires application. It features three input fields for file selection, each with a magnifying glass icon to the right. The 'Model' field contains 'C:\Users\wilso\Downloads\Example - Production Line\Production_line.cfg'. The 'Input' field contains 'C:\Users\wilso\Downloads\Example - Production Line\inputs.csv'. The 'Output' field contains 'C:\Users\wilso\Downloads\Example - Production Line\Production_line.dat'. At the bottom right, there are two blue buttons labeled 'Previous' and 'Next'.

Após a seleção dos arquivos, o OSIRES lê a primeira linha do arquivo .csv e determina a quantidade de variáveis que serão otimizadas. Neste caso são quatro, a quantidade de operadores de: corte tipo 1, corte tipo 2, dobra tipo 1 e dobra tipo 2.

Figura 11. Definição dos parâmetros das variáveis de otimização.

The screenshot shows the 'Step 02: Define resources bounds, steps and initial values' window in the Osires application. It displays two sections for defining parameters. The first section, 'OperatorsCut1', has four input fields: 'Minimum value' (1), 'Maximum value' (50), 'Step value' (1), and 'Initial value' (1). The second section, 'OperatorsCut2', has one visible input field for 'Minimum value' (1). At the bottom right, there are two blue buttons labeled 'Previous' and 'Next'.

O passo seguinte é a definição, se necessário, das restrições de aceitação para as respostas de otimização, as restrições do problema. Nesse caso, como o limite inferior para cada variável é igual a um, a primeira restrição

já é atendida de valor mínimo da soma das quatro variáveis igual a quatro, com nenhuma igual a zero.

Figura 12. Definição dos parâmetros das restrições das variáveis.

Osires

← Step 03: Define Constraints

[Simulation].OperatorsCut1
 [Simulation].OperatorsCut2
 [Simulation].OperatorsFold1
 [Simulation].OperatorsFold2
 this.SimTime/1[h]
 '[CutP].StateTimes("Working")/1[h]
 [CutQ1].AverageQueueTime/1[h]
 '[CutP2].StateTimes("Working")/1[h]
 [CutQ2].AverageQueueTime/1[h]
 '[FoldP].StateTimes("Working")/1[h]
 [FoldQ1].AverageQueueTime/1[h]
 '[FoldP2].StateTimes("Working")/1[h]
 [FoldQ2].AverageQueueTime/1[h]
 [PartSink].NumberProcessed*500-([Simulation].Operator...rkingTime/1[h]*0.05)-([FoldP2].WorkingTime/1[h]*0.05)

New:

Insert a new constraint +

#0 [Simulation].OperatorsCut1 + [Simulation].OperatorsCut2 <= 50 -

#1 [Simulation].OperatorsFold1 + [Simulation].OperatorsFold2 <= 50 -

Previous Next

A título de efeito didático, foram representadas duas restrições, onde as variáveis devem ter suas somas menores que 50. Para o presente problema, foi definido que a lotação máxima de espaço físico para a linha de produção seria de 100 funcionários.

Figura 13: Definição dos parâmetros do método de otimização

Osires

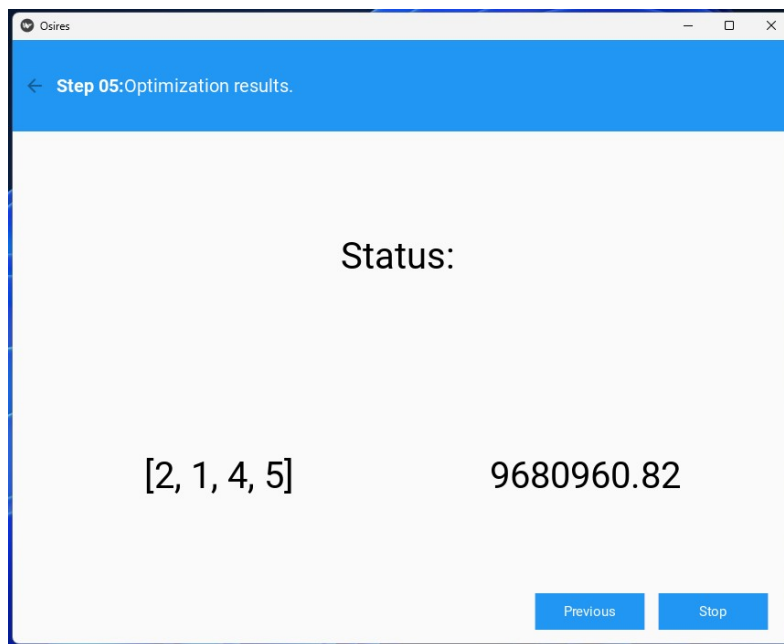
← Step 04: Define optimization parameters.

Objective function option: MAX

Select optimization method: GRASP

Save Back Run

Figura 14: Início do processo de otimização



7. Pontos de melhoria identificados

Até o momento de seu lançamento, para a versão 3.0, foram identificados os seguintes pontos para serem observados para a versão seguinte:

- 1) Ao se utilizar a função de minimização, pode ser gerado a apresentação de dois números para a última variável de otimização. Como medida de contenção inicial, pode ser criado uma restrição que impeça esses valores acima do limite máximo da variável de otimização. Este fator foi identificado de forma muito rara nos modelos testados.
- 2) Na entrada de dados, o OSIRES só aceita a alteração dos dados antes que o botão "Run" seja acionado. Depois da ação desse botão, todos os parâmetros ficam definidos assim como originalmente configurado nas telas anteriores.

8. Histórico de alterações e crédito das participações

OSIRES (ver.01)	
Atividades	Responsável
Idealização, gestão e manutenção do projeto	Wilson Trigueiro de S. Jr.
Definição do “front end”	Alex de Andrade Soares
Definição das telas de Reload, arquivos, parâmetros, restrições e otimização	Alex de Andrade Soares
Integração do Python com o JaamSim 2022-06, na realização das simulações.	Alex de Andrade Soares
Implementação do método “Scatter Search”	Alex de Andrade Soares
Implementação do método “Scatter Search + ML”	Alex de Andrade Soares
Implementação do método “GRASP”	Alex de Andrade Soares
Implementação do método “GRASP + ML”	Alex de Andrade Soares
Implementação de método para navegação entre os campos	Luís Arthur de Assis Moraes
Apresentação de exemplo no campo restrição	Luís Arthur de Assis Moraes
OSIRES (ver.02)	
Adição de método gráfico dinâmico, com a saída do resultado da otimização, por iteração.	Luís Arthur de Assis Moraes
Correção de erros de tratamento de dados de entrada de arquivos de simulação.	Luís Arthur de Assis Moraes
Atualização da versão do JaamSim utilizado para 2024-08.	Luís Arthur de Assis Moraes