Proceedings of the ASME 2016 35th International Conference on Ocean, Offshore and Arctic Engineering
OMAE2016
June 19-24, 2016, Busan, South Korea

OMAE2016-54274

MULTICRITERIA OPTIMIZATION FOR SYSTEM CONFIGURATION USING MONTE CARLO SIMULATION AND RAM ANALYSIS

Adriana Miralles Schleder
Naval Architecture and Ocean Engineering Department
University of São Paulo
São Paulo - Brazil
adrianamiralles@usp.br

Paula Cyrineu Araujo
Naval Architecture and Ocean Engineering Department
University of São Paulo
São Paulo - Brazil
adrianamiralles@usp.br

Marcelo Ramos Martins
Naval Architecture and Ocean Engineering Department
University of São Paulo
São Paulo - Brazil
mrmartin@usp.br

ABSTRACT

Currently, engineers should to deal with conflicting objectives, especially concerning safety and economics constraints. It is necessary to take into account performance indicators like reliability and availability coupled with economic criteria such as the costs of acquisition, maintenance and plant downtime. This paper aims at bringing up a rational process of selecting the optimal configuration of the system in the preliminary design phase considering these conflicting indicators.

Here, it is proposed a coupled approach using Monte Carlo Simulation and Genetic Algorithms to define the system configuration and the time interval between preventive maintenances in order to maximize the availability and the expected profit with the system operation considering possible constraints about the number of maintenance teams.

The approach proposed in this paper has shown to be promising for solving complex system design related to realistic scenarios in which conflicting performance and economic objectives must be taken into account.

INTRODUCTION

The complexity of the current engineering systems compel engineers to deal with conflicting objectives, specially concerning to safety and economics constrains. In order to optimize the design, one must take into account performance indicators like reliability and availability coupled with economic criteria such as the costs of acquisition, maintenance and plant downtime. The main goal of this paper is to present a rational process to select the optimal configuration of the system in the preliminary design phase considering these competing indicators.

For this nontrivial multi-objective optimization problem, there is no single solution that simultaneously optimizes each objective; there is a set of optimal solutions (possibly an infinite number) that can be represented in a Pareto front [1]. In this context, this paper proposes a coupled approach using Monte Carlo Simulation (MCS) and Genetic Algorithms (GA) to define the system configuration and the time interval between preventive maintenances in order to maximize the availability and the expected profit with the system operation considering possible constraints about the number of maintenance teams.

The Monte Carlo simulation is a widely used method in solving real engineering problems in several fields. Lately, the use of this method to determine the availability of complex systems and the monetary value of transactions and maintenance of production plants has increased. The complexity of current engineering systems coupled with the need of realistic considerations in shaping their availabilities and reliabilities result in very complex methods. Similarly, analyzes involving repairable systems and multiple additional events are very difficult to solve analytically.

As an alternative to such challenges, the Monte Carlo simulation simulates the system process and its random behavior from a computer model in order to create a realistic scenario over time and thus way predicted the variables interest. This method treats the problem as a number of real experiments conducted in a simulated time, estimating the probability and other parameters by counting the number of times that an event occurs in a simulation time. Input information are probability density functions (*pdf*) for fault and repair times and data about maintenance policies (intervals, duration and type among others)[2].

In this study, the MCS was used to estimate the reliability or availability of a given system configuration as well as its profit; different conditions of corrective and preventive maintenance policies were simulated and assessed during this stage. The algorithm, developed in the MATLAB environment, was validated using analytical results and data available in the literature; additionally, comparative analysis were performed comparing our simulations and simulations performed using thecommercial software MAROS [2].

In a second stage of the study, a genetic algorithm was developed to optimize the system configuration considering its availability and its profit function, which takes into account parameters such as the allocation of redundancies, the alternative types of components, the time interval between preventive maintenances and the number of available maintenance teams.

Genetic algorithms are search stochastic techniques based on natural selection and its usual form is described by Goldberg [4]. These algorithms are initialized with an arbitrary initial population of individuals called chromosomes representing a solution to the examined problem. A chromosome is a vector of parameters, which may take binary values or not. The chromosomes evolve through successive iterations, called generations. For each generation, the chromosome is evaluated using some fitness measures [5]. In order to create the next generation, i.e. children chromosomes, there are two options: cross two parents chromosomes of the current generation using a combination operator or modifying a chromosome with a mutation operator. A new generation is then formed by selecting the best chromosome and discarding those that have a lower fitness value, ensuring that the population will always remain with the same size. After several generations, the algorithms converge to the best chromosome or the best set of chromosomes when analyzing multiobjective problems. The GA proposed here was validated by the comparison between the best results achieved by the MCS and the Pareto optimal curve estimated by the optimization process.

The approach proposed in this paper has shown to be promising for solving complex system design related to realistic scenarios in which conflicting performance and economic objectives must be taken into account.

MCS ALGORITHM

We developed an algorithm in MATLAB environment that accomplishes the MCS for a generic system in series with n components; each component may have k redundancies in parallel, all active and capable of assuming 100% system load. It was not considered in this study the presence of redundancies in standby mode (which are passive redundancies that are inactive while the other component in parallel is working and only come into operation in case of failure of a component); this type of redundancy will be considered in future work.

The program inputs are the number of components n and the number of redundancies of each component k_n, which, in our case, the maximum value was set as 10 redundancies. For each component, C_{nk} (component k of the set of parallel redundancies n) it is built a matrix with the transition rates between the possible states of the components. It was assumed that all components have only one operating state defined as a state 1, and a fault state, set to 2. Therefore, the transitions are represented in a 2x2 array, in which the main diagonal contains only zeros, since the transition from one state to himself is not possible. This matrix contains the failure rate $\lambda_{1\rightarrow 2}$ and the repair rate $\lambda_{2\rightarrow 1}$, shown in Table 1. However, it is important to note that this matrix is applied to the case of the exponential distribution, which is represented by a single parameter λ. The Weibull distribution, for example, is represented by two parameters: a shape α and a scale β. In this case, the matrix must consider, for each transition, these two parameters. Similarly, if the component has transition times determined by other distributions, its array of transitions must contain the required parameters.

Table 1 - Transition matrix of the components

State	**1**	**2**
1	0	$\lambda_{1\rightarrow 2}$
2	$\lambda_{2\rightarrow 1}$	0

As inputs, mission time T_m of the system and the number of simulations n_{sim} must also be inserted. This number can vary greatly; even for simple systems, a high number of simulations is necessary to achieve results convergence, usually about 10^5 to 10^6 simulations.

The simulation starts with all active components in operating state 1. Then, the time of the next transition is calculated for each component. The C_{nk} component that presents the shortest transition time will be the one that will transition; thus, the simulation time counter t (which initially starts at zero) is replaced by this time of transition of the C_{nk} component. The transition time is randomly generated based on the failure and repair distributions.

After the first transition has been made, the process is repeated: the transition times are generated for all components, including the component that has just undergone a transition (as this is now in failure mode, the transition time generated for it is the time to repair). Once more, transition times generated are placed in ascending order and the shortest transition time is chosen.

The code considers two main situations: not taking into account corrective maintenances and the considering those. In the former case, when a failure occurs within the mission time, the code searches for some redundancy in operation. If there is a operational redundancy, the system still works; however, if redundancies are in failure mode, or if a failed component has no redundancy, the system goes into failure mode and the simulation ends. The number of system failures before the mission time and the total run is recorded; thus, the system reliability is estimated dividing the number of simulations that presented no system failures during the mission time by the total number of simulations. In the second case, where there are corrective maintenances, it is assumed that there are only perfect repairs (i.e. the component returns to its initial state of operation). The corrective maintenances may be applied by a limited or unlimited number of maintenance personnel; it was assumed that the same team could perform the maintenance of any component and there is the option to limit the number of available teams.

Even with an unlimited number of maintenance personnel, the system can go into failure mode (called downtime); this occurs because when there is a component failure that causes the system failure, it will remain in failure mode while the component is being repaired. In this case, the downtime variable receives the time during which the system was in failure mode, which may occur more than once during the simulation. In such case, the system availability is estimated dividing the total downtime by the total simulation time.

MCS algorithm validation

The MCS algorithm was validated in three steps: first, a simple case without maintenances for which the results could be achieved analytically; next, a case reported by Zio [6], in which only corrective maintenances are taken into account, was simulated and compared with the results reported; and finally, two systems, taking into account both preventive and corrective maintenances were simulated and compared with the results of the MAROS software [1]. Due to limited space, only the third case (the most complete) will be presented below.

To increase the reliability of complex systems, a possible alternative is the preventive maintenance program. Such a program can reduce the effects of component wear over time and has a significant impact on system life [6]. In the case of this study, it is assumed that the preventive maintenance restore the component to its original condition (as good as new); future work will take into account imperfect maintenances considering that the component is not restored to its original condition, which is a more realistic situation. Two different systems were tested, shown in Figures 1 and 2, their parameters are presented in Table 2 and Table 3.

The mission time for both systems was 10 years. The number of simulations used in MAROS software was 250 in both cases, and Monte Carlo algorithm used 250 simulations for the system 1, which is more complex, and 1000 simulations for the system 2, which does not require much computational effort. Table 4 and Table 5 present the results and the corresponding standard deviations obtained by MAROS and the MCS algorithm.

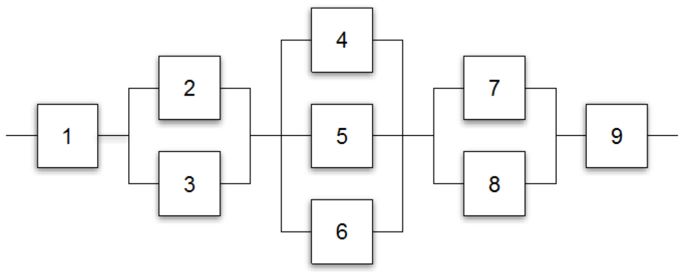

Figure 1- System 1

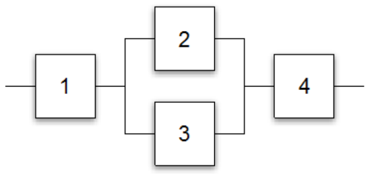

Figure 2 - System 2

Table 2 - System 1 parameters

Item	Failure dist.	β	α/λ	Repair dist.	λ (year)	Maintenance interval (year)
1	Exponential	-	4.38	Exponential	175.2	1.00
2	Weibull	1.2	0.5	Exponential	131.4	2.00
3	Weibull	1.2	0.5	Exponential	131.4	2.00
4	Weibull	2.4	0.25	Exponential	87.6	0.16
5	Weibull	2.4	0.25	Exponential	87.6	0.16
6	Weibull	2.4	0.25	Exponential	87.6	0.16
7	Weibull	1.4	0.7	Exponential	87.6	2.00
8	Weibull	1.4	0.7	Exponential	87.6	2.00
9	Exponential	-	4.38	Exponential	175.2	1.00

Table 3 - System 2 parameters

Item	Failure dist.	β	α/λ	Repair dist.	λ (year)	Maintenance interval (year)
1	Weibull	2.0	0.3	Exponential	175.2	1.0
2	Weibull	2.4	0.25	Exponential	87.6	2.0
3	Weibull	2.4	0.25	Exponential	87.6	2.0
4	Weibull	1.2	0.5	Exponential	43.8	1.5

Table 4 - Averaged availability estimated by MAROS and the MCS for system 1

Preventive maintenance	Teams	MAROS (%)	MCS (%)	Discre-pancy (%)	Simula-tion time (s)
Yes	1	85.970	95.23	10.77	73.00
Yes	2	92.687	95.60	3.14	74.03
Yes	3	91.829	94.65	3.08	77.93
Yes	∞	91.844	94.71	3.12	75.38
No	1	92.707	93.08	0.41	61.60
No	2	94.946	95.07	0.13	53.64
No	3	95.112	95.13	0.02	53.33
No	∞	95.116	95.21	0.10	53.31

Table 5 - Averaged availability estimated by MAROS and the MCS for system 2

Preventive maintenance	Teams	MAROS (%)	MCS (%)	Discre-pancy (%)	Simula-tion time (s)
Yes	1	1.58	93.72	4.03	50.54
Yes	2	1.31	94.47	3.43	46.90
Yes	3	1.31	94.43	3.27	48.91
Yes	∞	1.30	94.39	3.23	47.01
No	1	1.45	93.78	0.35	64.90
No	2	1.31	94.48	0.16	63.03
No	3	1.28	94.44	0.16	72.35
No	∞	1.28	94.49	0.11	62.47

It can be seen that the results are very similar in the cases without preventive maintenance; the mean discrepancy was about 0.18%. However, more significant discrepancies were found in relation to MAROS when the preventive maintenance was taken into account (4.26%). After further investigation of software functionality, it was verified that MAROS premises about preventive maintenance differs from premises implemented in the algorithm. The main differences are:

- The intervals for preventive maintenance of MAROS consider the overall time operating system while the algorithm considers the hour meter component; i.e. if preventive maintenance occurs every 100 hours, in MAROS the maintenances will occur at simulation time 100, 200, 300, etc. regardless of whether the component has failed in the meantime. In the algorithm implemented, the maintenance occurs every 100h of component operation; if the component fails before 100h, the hour meter resets and 100h of operation should be counted again before the next maintenance;

- In MAROS, when the preventive maintenance does not occur in the scheduled time by lack of staff the component goes into a queue until there is an available team. In the proposed algorithm, if the preventative maintenance is not possible at the scheduled time it will be made in the next scheduled preventive maintenance. For example, if it cannot happen at 100h of operation, it will occur at 200h of operation.

- Finally, our approach assumes that the preventive maintenance occurs only if the component is in an operation state. In MAROS, if the component is in failure mode, it will undergo the preventive maintenance as soon as it comes back to operating mode.

GENETIC ALGORITHM

As mentioned, in a second stage of the study, a GA, making use of the MCS approach previously presented, was developed (in the Matlab environment) in order to select the optimal configuration of the system in the preliminary design phase.

The parameters optimized by the algorithm proposed here are the number of redundancies of each component, the number of available maintenance crews and the interval between preventive maintenances of each component.

GA are search stochastic techniques based on natural selection; it operates on a set of chromosomes, called population, which is usually generated randomly. Once the algorithm advances in the new chromosome generation, the population wills increasingly tailored solutions to the goals of the problem, possibly converging on a solution curve.

Among the available GA models, the Nondominated Sorting Genetic Algorithm-II (NSGA-II) was implemented in our code; this method is widely used method for optimization of cases with multiple objectives [6]. This method selects the best candidates, classifying them into different sub-groups based on Pareto dominance relationships. In the NSGA-II method, the population of n individuals is initiated as usual and classified according to the Pareto front that each individual occupies. The first front (rank 1) consists of individuals who were not dominated by any other individual in the population. The second front (rank 2) includes individuals who are not dominated by any other individuals of the population with the exception of the first front individuals and so on. In addition to the rank assigned to these individuals, there is another parameter called crowding distance which must also be calculated. This parameter measures how close a individual is to his neighbors. A higher value of crowding distance will result in a greater diversity of the population. More details about this method are presented by Zio [6].

Once initialized the population and classified individuals, the parents chromosomes are selected from a selection (called binary tournament) which consists in restrict the population to a fixed number or a predefined percentage (called the size pool); and within this population choose the n best individuals (size tour) from its rank and of their crowding distance. An individual is selected if his rank is lower than another individual, or if the ranks are the same between the two, if your crowding distance is greater than the other individual. These parents chromosomes generate sons chromosomes from combination and mutation operators. Next, this new population, undergoes to a new classification of its individuals and only the best individuals are selected n, where n is the population size. The process continues until it reaches the desired number of generations.

The algorithm starts by determining the number of components n and the maximum number of redundancies for each component. Different types of components may be set with their respective parameters relating to the failure behavior and maintenance policy. The chromosomes used in this study have the format shown in Figure 3, where each gene represents a system parameter to be optimized.

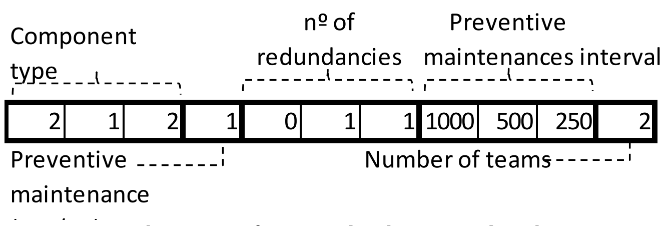

Figure 3 - Chromosome format used in the genetic algorithm

The initial population size n_{pop} is defined in a way that it depends directly on the total number of possible configurations. Next, n_{pop} chromosomes are randomly generated observing the maximum number of redundancies specified for each node. At this point, the Monte Carlo function is evaluated for each chromosome and the parameters required for the simulation are number of redundancies, type of each component, mission time and the number of simulations.

As the average availability and profit are computed for each component, the chromosomes are classified by their ranks of no dominance criteria and crowding distance. Subsequently, the number of generations is set; which defines the stopping criteria. In each generation the parent chromosomes must be chosen to be crossed, using the tournament selection method where the pool size parameter defines the percentage or number of chromosomes that will be selected randomly from the population and the tour size parameter defines the amount of chromosomes to be selected in this population. The best chromosomes are selected, comparing their rank and the crowding distance assigned to them.

Next, the probability of combination p_c is defined in order to determine whether the chosen parent chromosomes will be crossed or not. This probability is generally high, often being equal to 1 [7] and [8]. To achieve the combination of parent chromosomes, a random number varying from 1 to n-1 is generated and will determine the crossover point of the chromosomes. After combination and mutation operations, there is a new population increased by the sons chromosomes derived from combinations new chromosomes generated by mutation. In a nest step, the chromosomes are reclassified according to non-dominance and crowding distance criteria and only the best chromosomes are selected to the next phase. This process is repeated until the preset number of generations is attained. The result is defined by the chromosomes of the latest generation with rank 1;this set is the Pareto front representing the optimal solutions.

Genetic Algorithm validation

Two case studies were used to perform the validation of the proposed algorithm. The first seeks to optimize only redundancy allocation of a system with three components in series, assuming no preventive maintenances and a unlimited maintenance teams available to execute corrective maintenances. The second study case optimizes the component type, the use of redundancy, the preventive maintenance intervals and the number of maintenance personnel.

Case 1

In order to validate the algorithm described above, a system adapted from Zio [6] was used. It is a simple system with 3 nodes, each node can have no more of 4 components in parallel (redundancy equals 3), which generates a total of 64 possible configurations. All components present constant failure and repair rates, the number of maintenance personnel is unlimited and there is no preventive maintenance policy. In addition, all components are active and do not share load. At last, the corrective maintenances are perfect. Table 6 and Table 7 show the characteristics of the components and parameters used in the simulations.

Table 6 - Components features

Item	Failure rate (h^{-1})	Repair rate (h^{-1})	Acquisition cost ($)	Repair cost ($/$h^{-1}$)
A	$9.00x10^{-3}$	$4.0x10^{2}$	3000	30
B	$1.25x10^{-2}$	$1.0x10^{1}$	2000	30
C	$8.00x10^{-3}$	$2.0x10^{2}$	4000	35

Table 7 - System and simulation parameters

Parameter	Unit	Value
Plant yield	$	50000
Downtime cost	$/$h^{-1}$	50
Mission time	h	1000
Number of simulations	-	250
Initial population	-	10
Number of generations	-	6
Combination probability	-	1
Mutation probability	-	0.10

The objective function values are computed (by the MC algorithm presented in the previous section) for each one of the 64 settings and the solutions were classified as function of their ranks. Figure 4 shows the results highlighting the solutions with rank 1 (Pareto front – red points), which is the reference to assess the optimization process.

The optimization by the GA was simulated with an initial population of 20 individuals and 6 generations. Figure 5 shows the comparison of individuals with rank 1 found in the 6th generation of the genetic optimization algorithm and individuals with rank 1 (Pareto front) resulting from the scan of each of the 64 possible settings.

The settings whose values were rank 1 to the end of the simulation optimization represent a subset of the Pareto front found for the 64 settings. Additionally, there is a fast convergence to the optimal curve over the generations. It can be concluded that is an efficient algorithm to find the optimal solutions of the problem addressed. The simulation settings of all possible configurations lasted 3min10s and optimization 2 minutes.

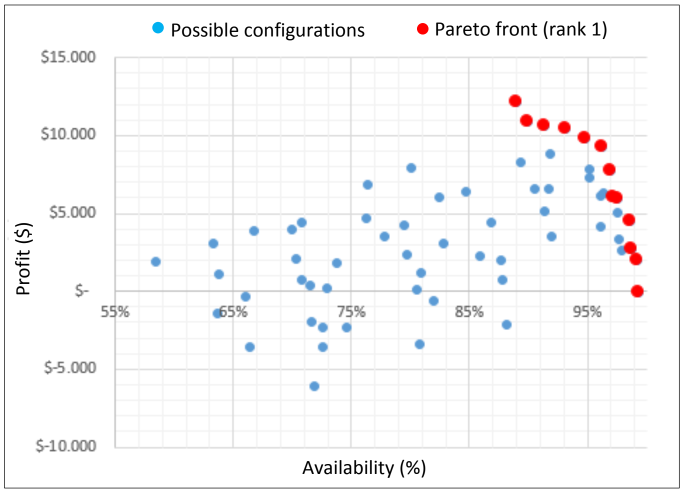

Figure 4 - Results of the 64 settings with emphasis on the Pareto front (solutions with rank 1)

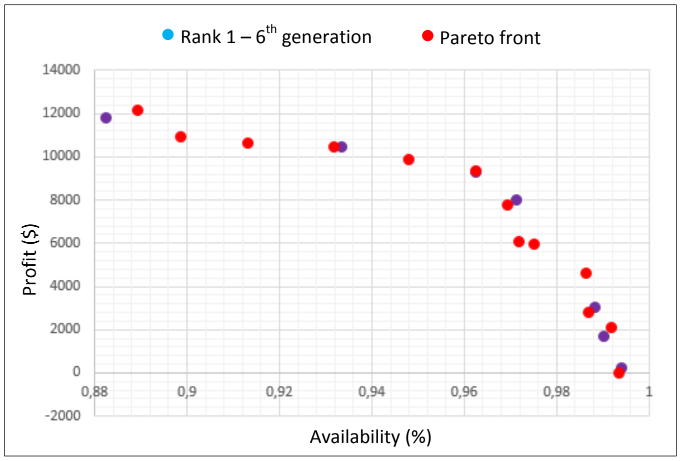

Figure 5 – Optimization results X Pareto front of all 64 settings (MCS)

Case 2

In the second study case, some complexities were added to optimization. In addition to the redundancy allocation, the optimization process should be able to define the best type of component to be used (with exponential failure distributions and Weibull), the intervals of preventive maintenances and the number of teams available for maintenance. Thus, a system with three components in series was selected; each component can have a redundancy or not and the first and third component may be of two different types (as presented in Table 8). Furthermore, one can choose whether the system will or not have preventive maintenances; there are two options of preventive maintenance intervals. Finally, it is possible to choose whether the system will have one, two or three maintenance crews.

It is worth to note that for the component type 1 there are no data to preventive maintenance; since, the failure distribution is exponential (without memory), thus the preventative maintenance is not effective. Table 9 shows the acquisition and maintenance costs of these components, and Table 10 shows the optimization general data.

Table 8 – Components features

	Component	1 Type A	1 Type B	2 Single type	3 Type A	3 Type B
Failure	**Distribution**	Exponen-tial	Exponen-tial	Weibull	Weibull	Weibull
	MTTF (h)	720	600	600	300	250
	λ or α	0.0014	0.0017	672	336	280
	β	–	–	3	3	2
Corrective maintenance	**Distribution**	Exponen-tial	Exponen-tial	Exponen-tial	Exponen-tial	Exponen-tial
	MTTR (h)	55	42	24	48	50
	λ	0.0182	0.0238	0.0417	0.0208	0.02
Preventive maintenance	**Distribution**	–	–	Exponen-tial	Exponen-tial	Exponen-tial
	Interval (h)	–	–	200 or 250	500 or 550	500 or 550
	λ	–	–	0.050	0.067	0.100
	MTTR (h)	–	–	20	15	10

Table 9 - Acquisition and maintenance costs

Component	Acquisition ($)	Corrective maintenance ($/h^{-1})	Preventive maintenance ($/h^{-1})
1 - Type A	6000	20	50
1 - Type B	5000	25	62
2 - Single type	7000	50	70
3 - Type A	6000	30	90
3 - Type B	5000	25	100

Table 10 - System features

Parameter	Value
Mission time (h)	1000
MCS	1000
Production ($)	100000
Downtime cost ($/h^{-1})	60
Team cost ($)	3000
Number of components	3
Maximum number of redundancies	1
Total of configurations	768

Initially, we used the MCS algorithm in all possible configurations, obtaining the points shown in Figure 6. The blue dots represent the results of all the simulated configurations and the red dots are for the non-dominated solutions (with rank 1). The simulation spent 52 minutes and 32 seconds.

Next, the GA is evaluated in order to achieve a curve of optimum configurations of the system taking into account the availability and the profit of the system.

Five updates were made in the simulations changing the size of the initial population, the number of generations and the probability of mutation. The probability combination of parent chromosomes was assumed equals 100% considering only one pivot point (arbitrarily chosen). Moreover, the mutation was made in a gene also arbitrarily chosen, respecting their range of possible values.

The first optimization generated a lot of suitable results to Pareto front, but only few solutions have been found. The second optimization provided many results; however, poorly adherent to the optimum curve. The third and fourth optimizations led to better results, but still some points far from the optimum curve. Finally, the last optimization had the most adherent results, nevertheless was also the most time consuming, 14 minutes of runtime simulation.

The five simulations showed that the number of non-dominated solutions usually is lower than 10, in contrast with the Pareto front that has 20 individuals. This occurs because the algorithm finds solutions close to each other, being difficult to increase the range of optimal solutions. Figure 6 presents comparison between the Pareto Front obtained by the MCS (the red points) and the optimal solutions achieved by the GA; it is possible to see how the GA solutions fit well in the Pareto front.

CONCLUSIONS AND FURTHER WORK

The implemented algorithms have proven useful and reliable. The results of validations for both the Monte Carlo code and for genetic algorithms were very satisfactory and adherents with analytical solutions, references and results of a market software.

In a general way, the algorithm has proven effective in converging to the optimum curve solutions. Although, it does not generate a very high number of solutions; one should take into account that the analysis system has been purposely designed to generate a wide spectrum of solutions in order to get a better view of the optimization process. Therefore, the convergence becomes more difficult once any small change in chromosome causes a sudden change in relation to the availability and profit. It can be said that this research is a solid foundation for further studies about multi-objective optimization, where larger and more complex systems can be analyzed. In future work, additional assumptions, such as standby redundancies or deteriorated state of the components, should be incorporated into the algorithms in order to provide a more realistic analysis.

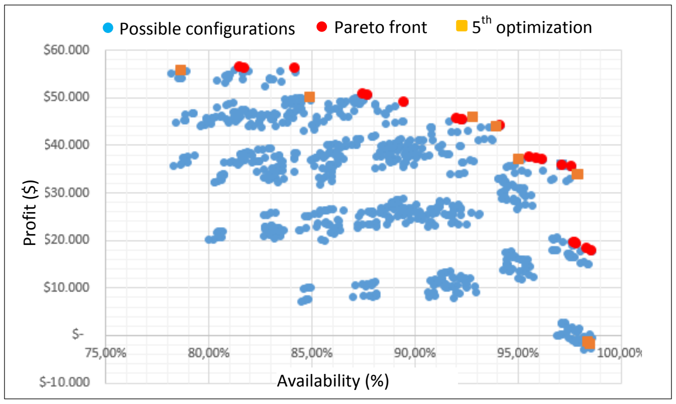

Figure 6 - Pareto front X Optimization

ACKNOWLEDGMENTS

The authors gratefully wish to acknowledge the Program for Development of Human Resources (PRH) from Petrobras and Brazilian National Petroleum. Natural Gas and Biofuels Agency (ANP) by the financial support.

REFERENCES

[1] CANTONI, M.; MASEGUERRA, M.; ZIO, E. Genetic algorithms and monte carlo simulation for optimal plant design. Reliability Engineering and System Safety, 2000.

[2] RAO, K. D. Dynamic fault tree analysis using monte carlo simulation in probailistic safety assesment. Relaibility Engineering and Safety System, 2009.

[3] MAROS Softwares - Monitoring and Remediation Optimization System. version 9. 2015. Det Norske Veritas GL.

[4] GOLDBERG, D. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison - Wesley, 1989.

[5] FOGEL, D.; GHOZEIL, A. Using fitness distributions to design more efficient evolutionary computations. In: Proceedings of IEEE International Conference on Evolutionary Computation, 1996. [S.l.: s.n.], 1996.

[6] ZIO. E. The Monte Carlo Simulation Method for System Reliability and Risk Analysis.[S.l.]: Springer-Verlag London. 2013.

[7] EBELING. C. E. An introduction to reliability and maintainability engineering. 2nd edition. ed. [S.l.: s.n.]. 2010

[8] MARSEGUERRA, M.; ZIO, E.; PODOFILLINI, L. Multiobjective spare part allocation by means of genetic algorithms and monte carlo simulation. Reliability Engineering and System Safety, 2004.