Otimização multicritério da configuração de sistemas utilizando análise RAM

Paula Cyrineu Araujo e Marcelo Ramos Martins
LabRisco - Laboratório de Análise, Avaliação e Gerenciamento de Riscos
Escola Politécnica da Universidade de São Paulo - Departamento de Engenharia Naval e Oceânica
Av. Prof. Mello Moraes, 2231 CEP 05508-030
Cidade Universitária - São Paulo-SP

1. INTRODUÇÃO

Ao se projetar uma planta de produção, deve-se levar em consideração não somente os custos associados como também a disponibilidade da operação. Isso implica que a análise dos níveis de confiabilidade, mantenabilidade e disponibilidade deve estar atrelada a um estudo econômico que estime, além dos custos de aquisição, os custos associados com o *downtime*, a manutenção e o reparo da planta de produção.

Assim sendo, várias decisões devem ser tomadas em relação ao tipo e número de componentes a serem utilizados na configuração do sistema. Na tentativa de se otimizar qualquer aspecto desta configuração, deve-se estar atento a muitos objetivos (baixos custos, altos rendimentos, alta confiabilidade, baixo risco de acidentes) que são geralmente conflitantes. Na literatura, várias abordagens já foram estudadas como o método dos gradientes, programação dinâmica, programação não linear e heurísticas [1]. Particularmente, o problema de alocação de redundâncias tem sido estudado extensivamente [2-4], sendo também foco deste trabalho.

Pelo ponto de vista teórico, a otimização da configuração do sistema equivale à maximização de uma função de mérito devidamente definida, que leva em conta a segurança do sistema e características econômicas. Mesmo para uma planta relativamente simples, a modelagem do seu comportamento de forma suficientemente realista resulta em função objetivo complexa, multivariável e não linear, o que impossibilita representá-la claramente de forma analítica. Esse aspecto tem as seguintes consequências: para uma produção desejada, há tantas possibilidades de configuração que o espaço de soluções para o algoritmo de otimização torna-se absurdamente grande; para uma determinada configuração, a avaliação da segurança e dos custos não é realizável através de métodos analíticos; e os métodos clássicos de otimização encontram enormes dificuldades no problema de maximização [5].

A melhor abordagem nesse caso é a que considera cada objetivo separadamente, visando identificar um conjunto de soluções que são equivalentes na ausência de uma hierarquia dos vários objetivos. Cada membro desse conjunto é melhor ou igual aos outros membros do conjunto no que se refere a alguns dos objetivos. Diferentemente da abordagem com apenas um objetivo que implica em baixa performance dos outros objetivos desejados, o conjunto identificado pela abordagem multicritério fornece um espectro de soluções "aceitáveis" entre as quais um meio-termo pode ser encontrado.

Neste contexto, apresenta-se a primeira fase de um trabalho que visa à otimização da alocação de redundâncias de um sistema pelo método dos algoritmos genéticos. Como passo inicial, foi desenvolvido um algoritmo que avalia a disponibilidade e custos médios de um sistema com uma configuração genérica a partir da simulação de Monte Carlo. Nessa simulação são contempladas tanto manutenções corretivas quanto preventivas, podendo-se limitar o número de equipes disponíveis.

2. OBJETIVOS DO TRABALHO

O projeto tem como objetivo desenvolver uma rotina capaz de otimizar a alocação de redundâncias de um sistema genérico, visando minimizar o custo e maximizar a sua disponibilidade e, consequentemente, a sua produção. Para tanto, dividiu-se o projeto em duas etapas: a primeira, cujos resultados são apresentados neste artigo, teve como objetivo desenvolver uma rotina que aplicasse a simulação de Monte Carlo em uma dada configuração de um sistema em série, cujos componentes podem ter ou não redundâncias. A simulação tem como objetivo calcular tanto a confiabilidade do sistema, no caso onde não há nenhum tipo de manutenção, quanto a sua disponibilidade média, para o caso onde há manutenção corretiva e/ou preventiva. Implementada a rotina, foi realizada uma série de validações para diversas configurações de sistema, utilizando-se como comparação tanto resultados encontrados na literatura, quanto resultados obtidos pelo software MAROS – *Monitoring and Remediation Optimization System.*

Em uma segunda etapa, será realizada a otimização multicritério da configuração do sistema pelo método de algoritmos genéticos, em que os parâmetros a serem otimizados são, principalmente, o número de redundâncias, o número de equipes de manutenção e os intervalos de manutenção preventiva. Para cada indivíduo gerado no processo de otimização será aplicada a rotina da primeira etapa, a fim de se ranquear os indivíduos que resultam em maior disponibilidade e menor custo.

3. DESCRIÇÃO DO TRABALHO REALIZADO

3.1 Algoritmo de simulação de Monte Carlo

Foi desenvolvido um algoritmo em linguagem Matlab que realiza a simulação de Monte Carlo para um sistema genérico com n componentes em série, sendo que cada componente pode apresentar k redundâncias em paralelo, todas ativas e capazes de assumir 100% da carga do sistema. Ou seja, não foi considerado neste trabalho redundâncias em *standby*, que são redundâncias passivas que ficam inativas enquanto outro componente em paralelo está funcionando e só entram em funcionamento no caso de falha de um componente.

Como entrada do programa, tem-se o número de componentes em série n e o número de redundâncias de cada componente k_n, cujo máximo valor está definido em 10 redundâncias. Para cada componente, C_{nk} (componente k do conjunto de redundâncias em paralelo do componente n), é inserida no programa uma matriz com as taxas de transição entre os possíveis estados. Foi considerado que todos os componentes só possuem um estado de funcionamento, definido como estado 1. Além do estado de funcionamento, cada componente tem o estado de falha, definido como 2. A matriz de transições, portanto, é uma matriz 2x2, com a sua diagonal principal contendo apenas zeros, já que a transição de um estado para ele mesmo não é considerada na simulação. Esta matriz contém a taxa de falha, $\lambda_{1\to2}$, e a taxa de reparo, $\lambda_{2\to1}$, como mostra a Tabela 1. Porém, é importante frisar que essa matriz se aplica para o caso da distribuição exponencial, que é representada por um parâmetro apenas, λ. A distribuição de Weibull, por exemplo, tem dois parâmetros que a define: um de forma, α, e um de escala, β. Neste caso, a matriz tem que contemplar, para cada transição, esses dois parâmetros. Da mesma forma, caso o componente possua tempos de transição determinados por outro tipo de distribuição, a sua matriz de transições deve conter os parâmetros necessários.

Tabela 1 - Matriz de transição dos componentes

Estado	1	2
1	0	$\lambda_{1\to2}$
2	$\lambda_{2\to1}$	0

Como entrada, também devem ser inseridos o tempo de missão, T_M, e o número de simulações, *nSim*. Esse número pode variar bastante, mas em geral, principalmente para os sistemas mais simples,

com menos componentes, é necessário um valor elevado de simulações para se conseguir uma convergência do resultado: na ordem de 5 a 6 casas decimais.

A simulação é iniciada com todos os componentes ativos, em estado de funcionamento 1. É, então, calculado para cada componente o tempo da próxima transição. No caso da primeira iteração, serão gerados os tempos em que ocorrerão as falhas dos componentes, já que todos estão neste primeiro momento em funcionamento. O componente C_{nk} que apresentar o menor tempo de transição é o que sofrerá essa transição, e dessa forma, o contador de tempo da simulação, t, que começa inicialmente com zero, será substituído por esse tempo de transição do componente C_{nk}.

A ideia é que o algoritmo de simulação funcione de forma independente a como estão definidas as matrizes de transições do componente. Ou seja, no código principal serão chamadas funções que devolvem o tempo de transição gerado de forma aleatória de acordo com os parâmetros da distribuição de falha e reparo do componente.

Após a primeira transição ter sido feita, o processo é repetido: são gerados os tempos de transição para todos os componentes, inclusive para o componente que acabou de sofrer a transição. Como este está agora em modo falha, o tempo de transição gerado para ele é o tempo de seu reparo. Novamente, são colocados os tempos de transição gerados em ordem crescente e é escolhido o menor tempo de transição. Ao contador de tempo de simulação é somado esse tempo, assim como ao tempo de funcionamento dos equipamentos que estão em funcionamento. Aqui é importante lembrar que, para distribuições exponenciais, não é necessário saber a quanto tempo um componente está funcionando para se gerar o seu tempo de transição, por se tratar de uma distribuição sem memória. Entretanto, como o código permite que os componentes tenham outras distribuições de tempo de falha e de reparo, como Weibull por exemplo, deve-se armazenar para cada componente o tempo que este está funcionando, pois esse tempo impactará no seu tempo de transição.

O código está dividido em duas vertentes: uma para o caso de não haver manutenção corretiva e uma para o caso onde há manutenção corretiva. Ou seja, no primeiro caso, quando há uma falha do componente, dentro do tempo de missão, o código procura se há alguma redundância em funcionamento. Se houver, o sistema continua funcionando, porém, se todas as redundâncias em paralelo estiverem no modo de falha, ou se um componente que falhou não possuir redundâncias, o sistema entra em modo de falha e a simulação termina. É contabilizado que houve falha antes do tempo de missão naquela simulação e o tempo total de funcionamento é também guardado. Para estes casos, calcula-se a confiabilidade do sistema, dividindo-se o número de simulações em que não houve falha do sistema antes de finalizado o tempo de missão pelo número total de simulações. O algoritmo da simulação sem manutenção é mostrado na Figura 1.

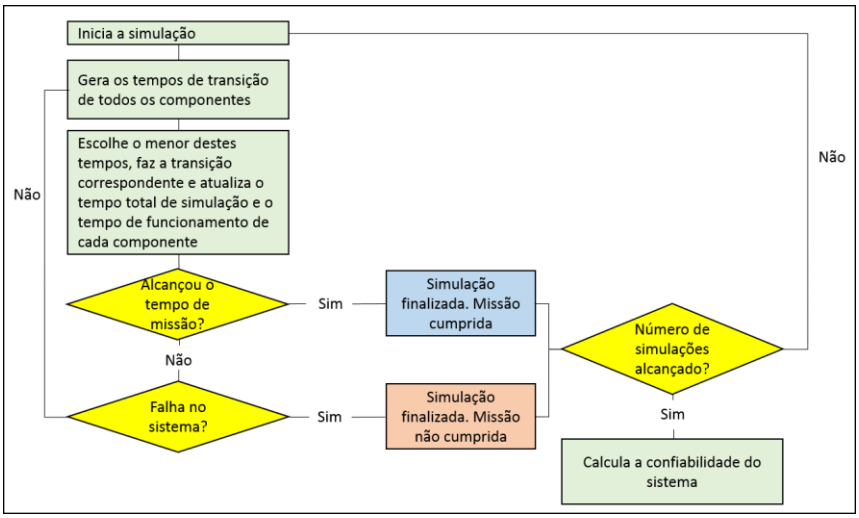

Figura 1- Fluxograma da simulação de Monte Carlo sem manutenção corretiva

No segundo caso, onde há manutenção corretiva, quando há falha em um componente, este é reparado e volta ao seu estado de funcionamento. Na prática, esse reparo costuma não ser perfeito, ou seja, o componente ao ser reparado não volta para o mesmo estado inicial, o que faz com quem a sua distribuição de tempo de falha se altere. Entretanto, para este trabalho, foram considerados apenas os reparos perfeitos, ou seja, aqueles que fazem o componente voltar ao seu estado inicial de funcionamento. Para o caso da manutenção corretiva, podem haver equipes de manutenção limitadas ou ilimitadas. Na prática, é comum os casos onde há equipes especializadas em um determinado tipo de componente. Para este trabalho foi considerada que uma mesma equipe pode fazer a manutenção de qualquer componente, tendo a possibilidade de se limitar o número de equipes disponíveis. Ou seja, se for definido como parâmetro de entrada que há apenas 3 equipes de manutenção disponíveis, caso haja três componentes em falha, o quarto componente a falhar não poderá ser imediatamente reparado. Será preciso que uma manutenção seja finalizada para que este componente comece a ser reparado.

Mesmo com um número ilimitado de equipes de manutenção, o sistema pode entrar em modo de falha, também chamado de *downtime*. Isso ocorre porque quando há a falha do componente que causa a falha do sistema, enquanto este componente estiver sendo reparado o sistema continua parado, mesmo com um número ilimitado de equipes. Nestes casos, a variável *downtime* recebe o tempo em que o sistema ficou em falha, que pode ocorrer mais de uma vez durante uma simulação. Ao fim da simulação, é calculada a não disponibilidade do sistema dividindo-se o tempo somado de *downtime* pelo tempo total de simulação. O fluxograma da simulação de Monte Carlo com manutenção corretiva é mostrado na Figura 2.

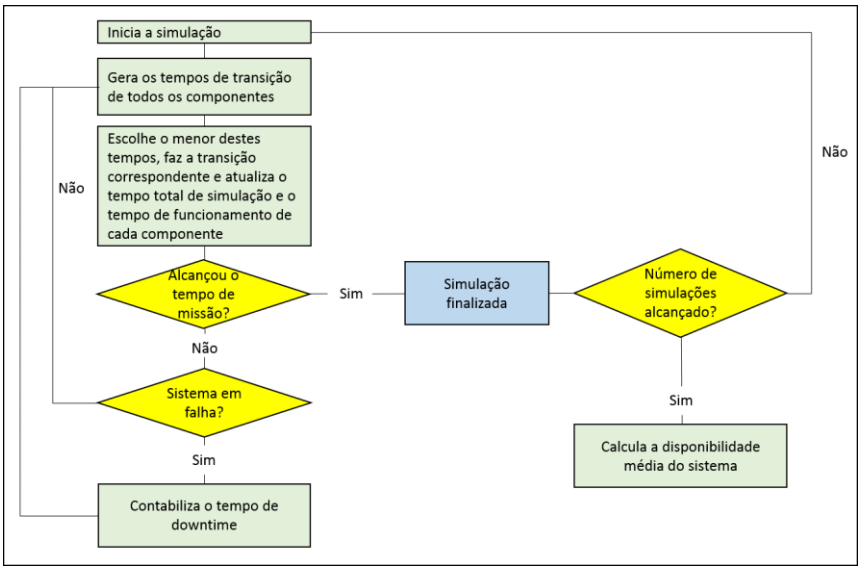

Figura 2 - Fluxograma da simulação de Monte Carlo com manutenção corretiva

Resumindo-se, para cada simulação são geradas aleatoriamente as transições do sistema conforme os parâmetros de distribuição dos tempos de transição de cada componente. Essas transições são feitas até que o contador de tempo da simulação alcance o tempo de missão, ou no caso onde não há manutenção corretiva, o sistema entre em modo de falha. Em ambos os casos, quando há falha em um componente, só haverá falha do sistema quando todas as redundâncias em paralelo deste componente estiverem também em modo de falha ou quando este componente não possuir nenhuma redundância.

3.2 Simulações

3.2.1 Estudo de caso 1: sistema simples

Depois de terminado o algoritmo da simulação de Monte Carlo, pesquisou-se em diversas referências algum estudo de caso simples que aplicasse a simulação de Monte Carlo em situações com e sem manutenção corretiva para que uma validação do algoritmo pudesse ser realizada. Infelizmente, as referências que trazem um caso de estudo neste contexto são poucas, e geralmente aplicam situações mais complexas que as que estão sendo usadas neste trabalho, como por exemplo, equipes diferenciadas para cada componente, degeneração do componente, manutenção imperfeita, entre outros.

Entretanto, a referência [6] apresenta um estudo de caso simples que possibilitou uma validação inicial do algoritmo. Trata-se de dois componentes em paralelo A e B ligados em série com um componente C, todos ativos. Todos os componentes possuem tempo de falha e de reparo com distribuições exponenciais. A esquematização deste sistema pode ser vista na Figura 3 e as matrizes de transição dos componentes podem ser vista na Tabela 2.

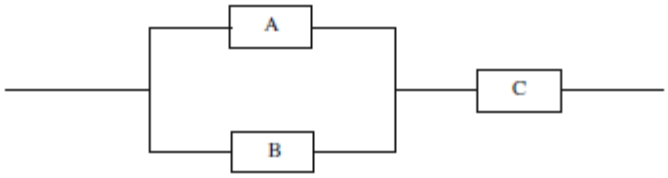

Figura 3 - Estudo de caso: sistema em série-paralelo [6]

Tabela 2- Matrizes de transição dos componentes A, B e C

Taxa de transição (h^{-1}) Componentes A e B		
	1	2
1	0	0.005
2	0.02	0
Componente C		
	1	2
1	0	0.0005
2	0.03	0

A validação para este estudo de caso foi dividida em duas partes: primeiramente, foi validado o código para a situação em que não há nenhum tipo de manutenção. Essa validação foi feita analiticamente, pelas fórmulas de confiabilidade de sistemas em série e em paralelo. Em um segundo momento, o código foi validado na situação de manutenção corretiva com equipes ilimitadas. Essa validação foi feita diretamente com os resultados apresentados em [6] e com os resultados obtidos pelo software MAROS versão 9.0.

3.2.2 Estudo de caso 2: sistemas complexos com manutenção preventiva

Para se aumentar a confiabilidade de sistemas complexos, uma alternativa possível é o programa de manutenção preventiva. Tal programa pode reduzir o efeito do desgaste do componente com o tempo e tem um impacto significativo na vida do sistema [7]. Para o caso deste trabalho, será considerado que as manutenções preventivas restauram o componente à sua condição original (*good as new*). Nessa etapa de validação foram testados dois sistemas mais complexos com diferentes distribuições de falha e de reparo, com e sem manutenção preventiva, limitando-se o número de equipes de manutenção disponíveis.

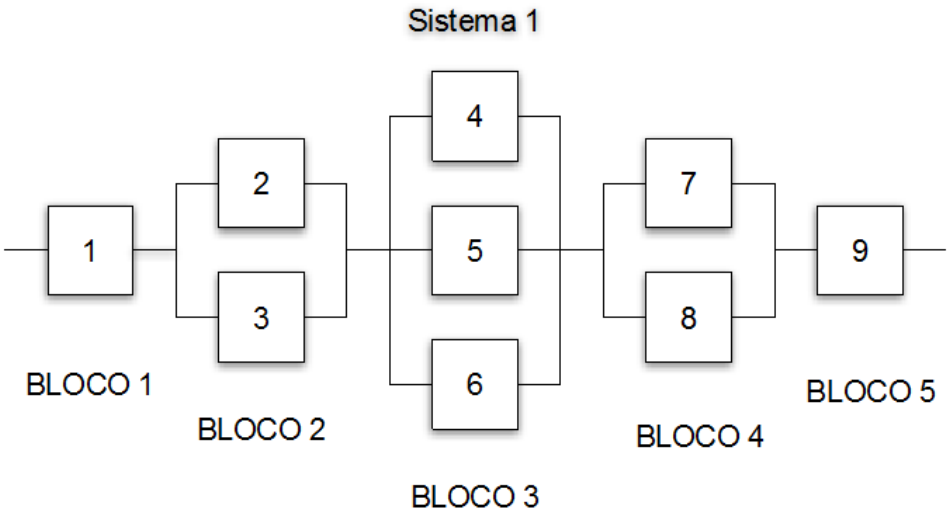

Figura 4- Esquematização do sistema 1

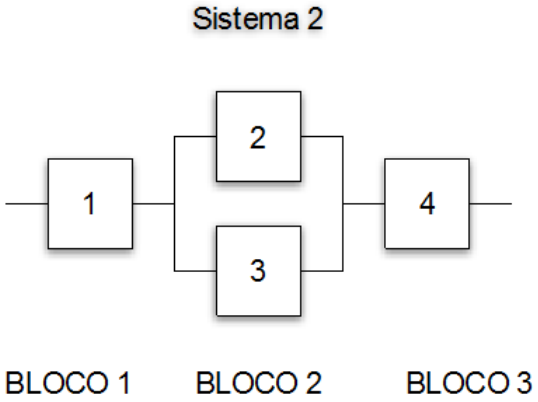

Figura 5 - Esquematização do sistema 2

As Figuras Figura 4 e Figura 5 esquematizam os dois sistemas utilizados, cujos parâmetros estão presentes nas Tabelas Tabela 3 e Tabela 4, respectivamente. Para as distribuições de falha dos componentes foram consideradas distribuições exponenciais e de Weibull, porém, a taxa de reparo foi modelada sempre como constante. É importante lembrar que, para um componente com taxa de falha constante, a manutenção preventiva não terá nenhum efeito sobre a confiabilidade do componente, devido à sua distribuição sem memória. Entretanto, ela pode acarretar maiores períodos de *donwtime* devido às interrupções que devem ser feitas nos componentes durante as manutenções preventivas.

Tabela 3 - Parâmetros do sistema 1

Bloco	Componente	Dist. de falha	β	α/λ	Dist. de reparo	λ (ano^{-1})	Intervalo entre manutenções preventivas (ano)
1	1	exponencial	-	4.38	exponencial	175.2	-
2	2	Weibull	1.2	0.5	exponencial	131.4	2
2	3	Weibull	1.2	0.5	exponencial	131.4	2

3	4	Weibull	2.4	0.25	exponencial	87.6	0.16
3	5	Weibull	2.4	0.25	exponencial	87.6	0.16
3	6	Weibull	2.4	0.25	exponencial	87.6	0.16
4	7	Weibull	1.4	0.7	exponencial	87.6	2
4	8	Weibull	1.4	0.7	exponencial	87.6	2
5	9	exponencial	-	4.38	exponencial	175.2	-

Tabela 4 - Parâmetros do sistema 2

Bloco	Componente	Dist. de falha	β	α/λ	Dist. de reparo	λ (ano^{-1})	Intervalo de manut. (ano)
1	1	Weibull	2.0	0.3	exponencial	175.2	1
2	2	Weibull	2.4	0.25	exponencial	87.6	2
2	3	Weibull	2.4	0.25	exponencial	87.6	2
3	4	Weibull	1.2	0.5	exponencial	43.8	1.5

Além disso, visando adicionar ainda mais complexidade à validação, foi limitado o número de equipes de manutenção disponíveis para 1, 2 ou 3 equipes. Neste caso, as equipes que fazem as manutenções corretivas são as mesmas que fazem as manutenções preventivas, e as taxas de reparo são também as mesmas para ambas as situações.

4. RESULTADOS OBTIDOS

4.1 Resultados caso de estudo 1

4.1.1 Sistema sem manutenção corretiva

Para este primeiro caso, o algoritmo calcula a não confiabilidade do sistema dividindo-se o número de simulações em que houve falha do sistema antes de terminado o tempo de missão pelo número total de simulações. A falha ocorre quando o componente A e B estão ao mesmo tempo em modo de falha ou quando o componente C entra em modo de falha. Assim que ocorre a falha do sistema, é terminada a simulação e a essa simulação é contabilizada uma falha.

Analiticamente, a confiabilidade do sistema neste caso pode ser facilmente calculada pela seguinte equação:

$$R_S(t) = \left[1 - \left(1 - R_A(t)\right)\left(1 - R_B(t)\right)\right]R_C(t) \quad (1)$$

onde $R_S(t)$ é a confiabilidade do sistema e $R_A(t)$, $R_B(t)$ e $R_C(t)$ são as confiabilidades dos componentes A, B e C, respectivamente.

Como se tratam de componentes com tempos de falha que seguem uma distribuição exponencial, temos:

$$R_S(t) = \left[1 - \left(1 - e^{-\lambda^A_{1\to2}t}\right)\left(1 - e^{-\lambda^B_{1\to2}t}\right)\right] e^{-\lambda^C_{1\to2}t} \quad (2)$$

onde $\lambda^i_{1\to2}$ é a taxa de falha do componente i.

O algoritmo foi testado para diversos tempos de missão, variando-se o número de simulações e os resultados estão apresentados nas Tabelas Tabela 5 a Tabela 8.

Tabela 5 - Resultados para tempo de missão de 10h

Tempo de missão = 10h			
Nº de simulações	Resultado analítico	Resultado da simulação	Erro (%)
1e3	99,26%	99,30%	0,04%
5e3	99,26%	99,10%	0,16%
1e4	99,26%	99,37%	0,11%
5e4	99,26%	99,28%	0,02%
1e5	99,26%	99,30%	0,04%
5e5	99,26%	99,27%	0,01%

Tabela 6 - Resultados para tempo de missão de 100h

Tempo de missão = 100h			
Nº de simulações	Resultado analítico	Resultado da simulação	Erro (%)
1e3	80,40%	81,70%	1,62%
5e3	80,40%	80,60%	0,25%
1e4	80,40%	81,10%	0,87%
5e4	80,40%	80,44%	0,05%
1e5	80,40%	80,27%	0,16%
2e5	80,40%	80,40%	0,00%

Tabela 7 - Resultados para tempo de missão de 500h

Tempo de missão = 500h			
Nº de simulações	Resultado analítico	Resultado da simulação	Erro (%)
1e3	12,26%	11,10%	9,46%
5e3	12,26%	12,28%	0,16%
1e4	12,26%	11,97%	2,36%
5e4	12,26%	12,15%	0,90%
1e5	12,26%	12,21%	0,41%
2e5	12,26%	12,25%	0,08%

Tabela 8 - Resultados para tempo de missão de 1000h

Tempo de missão = 500h			
Nº de simulações	Resultado analítico	Resultado da simulação	Erro (%)
1e3	0,81%	1,00%	23,46%
5e3	0,81%	0,84%	3,70%
1e4	0,81%	0,70%	13,58%
5e4	0,81%	0,84%	3,70%
1e5	0,81%	0,81%	0,00%
2e5	0,81%	0,82%	1,23%

Pode-se verificar pelos resultados que as simulações com tempos de missão menores geram resultados mais apurados, ou seja com menores erros. Também se verifica nos quatro casos que aumentar o número de simulações não ocasiona necessariamente uma diminuição do erro. As soluções obtidas pelas simulações tendem a oscilar em volta da solução analítica e, para tempos de missão maiores, é necessário um maior número de simulações para se alcançar convergência.

4.1.2 Sistema com manutenção corretiva

Nesta situação, assim que um componente falha, uma equipe de manutenção começa imediatamente a reparar esse componente. Seguindo as mesmas características do estudo de caso apresentado em [6], tem-se para essa simulação:

- Os componentes são binários, com um estado de funcionamento (estado 1) e um estado de falha (estado 2)
- O tempo de transição do componente *i* do estado j_i para o estado m_i é uma variável estocástica que segue uma distribuição exponencial com taxa constante $\lambda^{i}_{j_i \to m_i}$, *i*=1, 2, 3 e j_i,m_i=1,2
- Não há dependências de nenhum tipo (carregamento, *standby*, etc.)
- O reparo começa imediatamente após a falha do componente
- Reparos são feitos sempre com sucesso e levam o componente a uma condição perfeita de funcionamento (e.g., *as good as new*)

O tempo de missão foi considerado como sendo de até 1000h e as taxas de transição dos componentes são as mesmas apresentadas na Tabela 2. Para o MAROS foram utilizadas 1e3 simulações e para o algoritmo de Monte Carlo foram feitas duas diferentes simulações para cada tempo de missão, uma com 1e5 simulações e outra com 5e5 simulações. Os resultados das simulações podem ser vistos na Tabela 9.

Tabela 9 - Resultados de disponibilidade média (%) encontrada para o software MAROS e para o algoritmo de Monte Carlo

Tempo de missão (h)	MAROS 1e3 simulações	Monte Carlo 1e5 simulações	Erro (%)	Monte Carlo 5e5 simulações	Erro (%)
20	99,38%	99,36%	0,02%	99,38%	0,00%
40	98,48%	98,69%	0,22%	98,69%	0,22%
60	97,93%	98,03%	0,11%	98,05%	0,13%
80	97,42%	97,61%	0,19%	97,58%	0,16%
100	96,90%	97,20%	0,30%	97,19%	0,30%
200	95,64%	96,12%	0,50%	96,10%	0,47%
300	95,13%	95,68%	0,58%	95,66%	0,56%
400	95,10%	95,45%	0,37%	95,43%	0,35%
500	94,92%	95,30%	0,40%	95,32%	0,42%
700	94,82%	95,17%	0,37%	95,16%	0,35%
800	94,80%	95,11%	0,33%	95,10%	0,31%
900	94,70%	95,06%	0,37%	95,09%	0,40%
1000	94,69%	95,06%	0,39%	95,05%	0,39%

Graficamente, podemos comparar os resultados obtidos tanto pelo MAROS quanto pelo algoritmo de Monte Carlo, Figura 6, com os resultados obtidos por [6] mostrados na Figura 7.

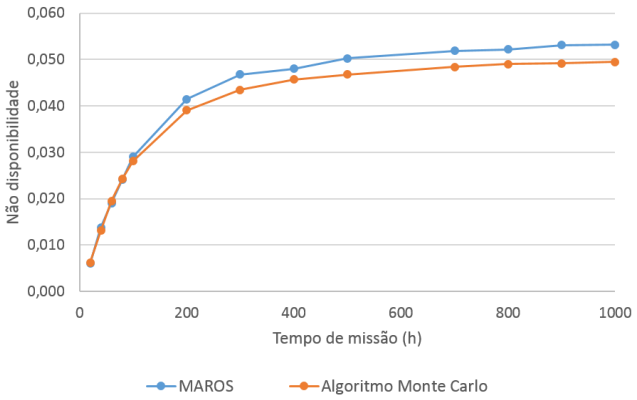

Figura 6 - Comparação entre os resultados de não disponibilidade obtidos pelo MAROS e pelo algoritmo de Monte Carlo com 5e5 simulações para o caso com manutenção corretiva

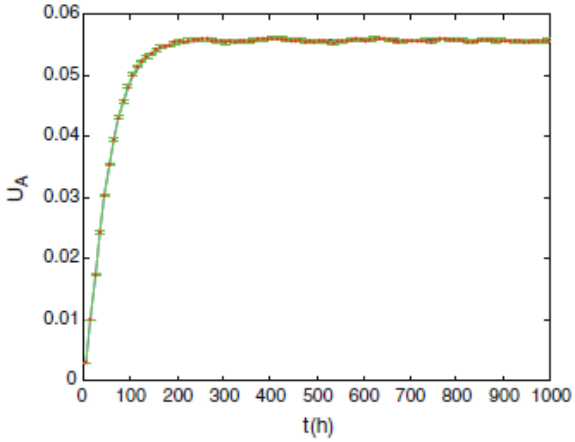

Figura 7 - Resultados de não disponibilidade para o caso de estudo com manutenção corretiva [6]

É possível verificar que os resultados obtidos pelo algoritmo de simulação de Monte Carlo estão muito próximos pelos valores encontrados pelo software MAROS. Percebe-se também que aumentar o número de simulações de 1e5 para 5e5 não necessariamente melhora o erro do resultado, porém aumenta consideravelmente o tempo de simulação. No caso de tempo de missão de 1000h, por exemplo, o tempo total de 1e5 simulações é de 120 segundos, sendo que para 5e5 simulações o tempo total é de 792 segundos, resultando em mais de 13 minutos de simulação.

Observando-se o gráfico apresentado na Figura 7, os resultados obtidos pelo algoritmo se distanciam um pouco mais, se comparados com os resultados do MAROS. Porém, ainda pode se considerar que os resultados estão satisfatoriamente próximos do desejado e que o algoritmo da simulação de Monte Carlo foi implementado com sucesso.

4.2 Resultados do estudo de caso 2

O tempo de missão para os dois sistemas é de 10 anos. O número de simulações utilizadas no software MAROS é de 250 para todos os casos, e no algoritmo de Monte Carlo foi utilizada 250 simulações para o sistema 1, que é mais complexo, e 1000 simulações para o sistema 2 que não demanda tanto esforço computacional. Nas Tabelas Tabela 10 e Tabela 11 podem ser vistos os resultados das

simulações e os desvios-padrão correspondentes obtidos pelo MAROS e o resultado obtido pelo algoritmo de Monte Carlo, com sua diferença em relação ao MAROS e seu tempo total de simulação.

Tabela 10- Resultados das simulações para o sistema 1

Manutenção preventiva	Equipes de manutenção	Disponibilidade MAROS (%)	Desvio-padrão MAROS (%)	Disponibilidade Monte Carlo (%)	Diferença (%)	Tempo de simulação (s)
Sim	1	85,97%	2,16%	93,38%	3,65%	214,44
Sim	2	92,69%	1,00%	95,80%	4,89%	222,34
Sim	3	91,83%	0,85%	95,99%	4,97%	222,54
Sim	Ilimitada	91,84%	0,74%	95,99%	4,97%	219,68
Não	1	92,71%	1,13%	93,89%	1,89%	168,53
Não	2	94,95%	0,74%	95,66%	2,66%	152,69
Não	3	95,11%	0,74%	95,78%	2,77%	153,08
Não	Ilimitada	95,12%	0,75%	95,69%	2,68%	154,48

Tabela 11 - Resultados das simulações para o sistema 2

Manutenção preventiva	Equipes de manutenção	Disponibilidade MAROS (%)	Desvio-padrão MAROS (%)	Disponibilidade Monte Carlo (%)	Diferença (%)	Tempo de simulação (s)
Sim	1	90,09%	1,58%	93,72%	4,02%	192,98
Sim	2	91,34%	1,31%	94,47%	3,43%	201,65
Sim	3	91,44%	1,31%	94,43%	3,26%	203,98
Sim	Ilimitada	91,44%	1,30%	94,39%	3,23%	197,86
Não	1	92,15%	1,45%	93,78%	1,76%	184,77
Não	2	93,18%	1,31%	94,48%	1,40%	175,20
Não	3	93,20%	1,28%	94,44%	1,34%	176,47
Não	Ilimitada	93,20%	1,28%	94,49%	1,39%	178,22

Pode-se verificar a partir dos resultados que para o algoritmo de Monte Carlo considerando a manutenção preventiva não gerou diferença nos resultados de disponibilidade, causando diferenças maiores em relação ao software MAROS. Essa diferença pode ser devido ao fato de que o algoritmo de Monte Carlo utiliza a simulação direta, com o método da transição do tempo mínimo, enquanto que é possível que o software MAROS utilize o método indireto, com pequenos *time steps*. Uma outra possibilidade é que, diferentemente do algoritmo que considera que o horímetro do componente zera assim que ele sofre uma falha ou manutenção preventiva, o software MAROS talvez considere que o horímetro do componente é ininterrupto, o que causará diferença nos resultados pois as manutenções preventivas ocorrerão em momentos diferentes. Em uma próxima etapa, será verificado com mais afinco o porquê dessa diferença e uma nova simulação poderá ser executada.

5. CONCLUSÕES E PRÓXIMOS PASSOS

Este artigo apresenta os resultados obtidos na primeira fase do projeto ainda em desenvolvimento. Durante essa primeira etapa do trabalho, uma revisão bibliográfica extensa sobre os temas de confiabilidade, de risco e simulação de Monte Carlo foi realizada, o que permitiu uma implementação robusta do algoritmo de simulação de disponibilidade de sistemas.

O código se mostrou confiável para simulações com e sem manutenção corretiva, podendo se limitar ou não as equipes de manutenção. Foram testados diferentes sistemas, com diferentes distribuições de falha e reparo, e os valores se mostraram satisfatórios. As simulações com manutenção preventiva, por outro lado, mostraram alguns desafios em relação à interferência da falha do sistema na manutenção

preventiva dos componentes. Em um próximo passo, esse ponto será trabalhado para uma melhor análise e entendimento das diferenças entre as simulações do algoritmo de Monte Carlo e do software MAROS.

Com a etapa de implementação do algoritmo de Monte Carlo finalizada, a próxima etapa consiste no estudo do método de algoritmos genéticos que possibilitará a otimização da configuração dos sistemas analisados. Ou seja, com esse método será possível descobrir qual configuração (número de redundâncias, número de equipes, intervalo entre as manutenções preventivas) resulta em maior disponibilidade e menor custo.

6. REFERÊNCIAS

[1] F. Tillman, C. Hwang e W. Kuo, "Optimization techniques for system reliability," *IEEE Transactions on Reliability*, Vols. %1 de %2R-26, n. 3, pp. 148-155, 1977.

[2] R. Bellman e S. Dreyfus, "Dynamic Programming and the Reliability of Multicomponent Devices," *Operations Reserach*, vol. 6, n. 2, pp. 200-206, 1958.

[3] M. Chern, "On the computational complexity of reliability redundancy allocation in a series system," *Operations Research Letter*, pp. 309-315, 1992.

[4] Fyffe, D. E, W. W. Hines e N. K. Lee, "System Reliability Allocation and a Computational Algorithm," *IEEE Transactions on Reliability*, pp. 64-69, 1968.

[5] M. Cantoni, M. Marseguerra e E. Zio, "Genetic algorithms and monte carlo simulation for optimal plant design," *Reliability Engineering & System Safety*, vol. 68, n. 1, pp. 29-38, 2000.

[6] E. Zio, The Monte Carlo Simulation Method for System Reliability and Risk Analysis, Springer Series in Reliability Engineering, 2013.

[7] C. Ebeling, An Introduction to Reliability and Maintainability Engineering, 2010.