Otimização multicritério de configurações de sistemas mecânicos utilizando analise RAM

Ruben Benites Perez – ruben.benites@usp.br
Marcelo Ramos Martins – mrmartin@usp.br
Eduardo Alexandre Bonetto Ferreira – Eduardo_bferreira@hotmail.com
Everton Nogueira Lima – evertonlm88@usp.br
LabRisco - Laboratório de Análise, Avaliação e Gerenciamento de Riscos
Departamento de Engenharia Naval e Oceânica, Escola Politécnica, Universidade de São Paulo

RESUMO

Este artigo apresenta um protótipo de um software para otimizar sistemas mecânicos nas fases de projeto. O software está sendo desenvolvido sob a filosofia de Programação Orientada a Objetos em C++, seguindo a metodologia da análise RAM (*Reliability, Availability and Maintainability*) e utilizando diagrama de blocos de confiabilidade para modelar o sistema e o método de simulação de Monte Carlo para avaliar suas funções de mérito. Para a otimização da configuração do sistema e de seus parâmetro interesse foi utilizada como metodologia, uma Meta-Heurística por meio da técnica dos Algoritmos Genéticos.

INTRODUÇÃO

A busca por melhorias nos processos de engenharia além de ser uma área em constante desenvolvimento é um desafio comum e obrigatório do dia a dia dos engenheiros, em especial na fase de concepção de novos sistemas; melhorar estes sistemas significa muitas vezes ter que otimizá-los, tendo em conta vários critérios simultaneamente, sendo que, na maioria das vezes, estes critérios são conflitantes entre si. Este tipo de problema implica não só no incremento da dificuldade na busca da solução ótima, mas também no fato de que raras vezes uma dada solução será a melhor sob todos os critérios considerados. Nestes casos a solução ótima passa a ser relativa e será necessário um esforço adicional da equipe de projeto para escolher uma entre as várias soluções ótimas.

A situação descrita motivou o grupo de pesquisa do Laboratório de Análise, Avaliação e Gerenciamento de Risco (LabRisco) a projetar e desenvolver uma ferramenta computacional que apoie os engenheiros na escolha da melhor configuração do sistema desejado na fase de concepção, do projeto. A partir dos requisitos e restrições fornecidos pelo usuário, o software procurará o conjunto das configurações mais eficientes do sistema, respeitando sempre as restrições exigidas pelo usuário. Pela natureza combinatória do problema, as possíveis configurações aumentam exponencialmente na medida em que é adicionado um novo item ao sistema. Desta forma, a análise de cada uma das possíveis configurações torna-se computacionalmente inviável, pois para cada configuração do sistema deveria ser feita uma análise das variáveis: confiabilidade, mantenabilidade, disponibilidade, e custo. São consideradas ainda restrições como, por exemplo, a disponibilidade de componentes no mercado, número limitado da equipe para a realização de operações de manutenção do sistema, entre outras. Estas variáveis são do tipo estocástico, já que dependem de variáveis aleatórias (v.a.) como o tempo de falha e o tempo de reparo. Para estimá-las é utilizado o método de Simulação de Monte Carlo (Monte Carlo Simulation MCS) [1], mas a simulação de todas as configurações do sistema derivadas de todas as combinações possíveis, demandaria um custo computacional muito alto. Porém, o método de MCS pode ser aplicado eficientemente às pequenas porções de todas as possíveis soluções. A questão agora é: quais poderiam ser estas pequenas porções?

Para tratar este complexo problema de otimização, no software OtimizLabRisco implementou-se uma inovadora metodologia apresentada em [2] chamada de Meta-heurística que está sendo amplamente utilizada. Dentro da Meta-heurística existe uma família de algoritmos, denominados de Algoritmos Evolutivos, e estes por sua vez, possuem um grupo denominado Algoritmo Genético (*Genetic Algorithm GA*).

O conceito de GA foi apresentado por [3]. Este método é baseado originalmente em uma analogia biológica. Como é sabido da origem das espécies, tanto as plantas, como os animais e outros organismos, se valem de processos de seleção natural com base em características genotípicas, as quais estão determinadas diretamente em seu material genético, que são definidas de acordo com a expressão de sequências padrões (genes) presentes nos cromossomos herdados dos pais. Dessa forma, as populações dos organismos são

compostas por diversas sequências, relacionadas com a segregação de determinados genes. Além deste processo, dentro dos próprios genes podem ocorrer também variações genéticas (mutações) que levam a trocas aleatórias dos valores presentes nestas sequências, levando ainda a uma melhor adaptação das condições do ambiente. Esta analogia genética então permite a obtenção de combinações aleatórias de um conjunto de dados, gerando alternativas para resolver problemas numéricos.

OBJETIVO

O objetivo deste trabalho é mostrar as vantagens de utilizar GAs como motor de busca de máximos em um problema de domínio combinatório de grandes dimensões e alta complexidade, e sua implementação em uma linguagem de Programação Orientada a Objetos (Object-Oriented Programming OOP) como é C++.

DESCRIÇÃO DO TRABALHO REALIZADO

O presente trabalho esta dividido nas seguintes etapas: (i) Análise RAM; (ii) Algoritmo Genético; (iii) Protótipo do software OptimizLabRisco; (iv) Caso de estudo; (v) resultados; (vi) conclusão e comentários finais.

ANÁLISE RAM

A análise RAM de um sistema envolve o estudo da confiabilidade, da disponibilidade e da manutenabilidade com o intuito de avaliar seu desempenho e obter informação que possa ser útil para melhorar sua produtividade. Com os resultados das análises é possível identificar componentes críticos para o bom desempenho do sistema e poder tomar decisões para reduzir o impacto negativo das falhas.

A análise RAM é um campo relativamente novo, que surgiu com o aumento da complexidade dos sistemas. Diferente da abordagem tradicional da engenharia, que utiliza modelos determinísticos e agrega fatores de segurança no projeto para garantir uma confiabilidade mínima do sistema, a análise RAM utiliza um modelo estocástico baseado no fato de que os tempos de falhas e os tempos de reparo são v.a.. Assim, ela busca descrever o sistema de uma forma mais real.

A primeira etapa da análise RAM é o levantamento de dados a respeito do sistema, comportamento histórico de falhas e de tempos de reparo para os diferentes modos de falha dos componentes.

O segundo passo trata da modelagem da configuração do sistema através de diagramas que ilustram as conexões e dependências entre os componentes, além dos melhores modelos de distribuição de falha e reparos para cada componente, e também das restrições nos recursos disponíveis para a manutenção, chamados de RBD (*Reliability Block Diagrams*) ou Diagramas de Blocos de Confiabilidade

Finalmente, é realizada uma simulação do tempo de missão definido para o sistema.
A continuação serão apresentados os conceitos básicos e seus respetivos modelos matemáticos relacionados à analise RAM [4].

Confiabilidade do componente

Probabilidade de um item, componente ou sistema executar a função para a qual foi projetado durante um período de tempo pré-definido, sob determinadas condições ambientais e operacionais, dado que estava operando ou em condições de operar no instante inicial. Matematicamente, a Confiabilidade R(t) é representada pela equação (1)

$$R(t) = \Pr(T \geq t | c_1, c_2 \dots) \quad (1)$$

$Onde$:

$t \coloneqq Tempo\ de\ miss\tilde{a}o.$
$T \coloneqq Variável\ aleatória\ que\ representa\ o\ instante\ de\ falha.$
$c_1, c_2 \dots \coloneqq São\ as\ condições\ de\ operação.$

Disponibilidade do componente

Probabilidade de que o sistema seja capaz de executar, em um dado instante e sob as condições ambientais e operacionais especificadas, a função para qual foi projetado. Dessa forma, a disponibilidade pode ser interpretada como sendo a ração entre o tempo em que o sistema está em modo operacional e o tempo de missão. Matematicamente, a disponibilidade A(t) pode ser representada pela equação (2)

$$A(t) = \frac{t_{operacional}}{t_{operacional} + t_{falha}} \quad (2)$$

$Onde$:

$t_{operacional} \coloneqq Tempo\ que\ o\ sistema\ esteve\ operacional$
$t_{falha} \coloneqq Tempo\ que\ o\ sistema\ esteve\ não\ operacional.$
$t_{operacional} + t_{falha} = t_{missão}$

Mantenabilidade do componente

Probabilidade de que um componente em estado de falha seja reparado à sua condição original dentro de um intervalo de tempo especificado, sendo a manutenção realizada seguindo-se procedimentos pré-estabelecidos e sob condições especificadas. Matematicamente, a mantenabilidade M(t) é representada pela equação (3)

$$M(t) = \Pr(T \leq t | c_1, c_2 \ldots) \quad (3)$$

$Onde$:

$t \coloneqq Tempo\ de\ reparo\ que\ deve\ ser\ especificado.$
$T \coloneqq v.a.\ que\ representa\ o\ tempo\ requerido\ para\ o\ reparo.$
$c_1, c_2 \ldots \coloneqq Condições\ sob\ as\ quais\ foi\ realizado\ o\ reparo.$

Falha: Interrupção da capacidade do sistema em desempenhar a função para qual foi projetado.

Falta: Representa o estado no qual o sistema está inapto para realizar sua função.

Confiabilidade de sistemas

Na maioria dos sistemas deseja-se conhecer a confiabilidade de sistemas complexos, constituídos de múltiplos componentes. Assim, deve-se estudar também como a relação entre os componentes influencia o funcionamento de todo o sistema. Neste trabalho serão estudados os sistemas configurados em série com a possibilidade de componentes em redundância.

Configuração em serie.

Na configuração em serie na falha de um componente ocasiona a falha de todo o sistema, ou seja, todos os componentes são considerados críticos. Sua representação em diagrama de blocos é ilustrada na Figura 1.

Assumindo que os modos de falhas dos componentes são independentes, a equação que representa a confiabilidade do sistema é,

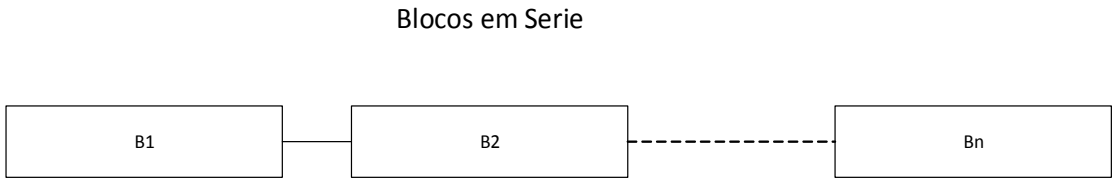

Figura 1 – Blocos em serie

$$R_s(t) = R_1(t)R_2(t) \dots R_n(t) \leq \min(R_1(t)R_2(t) \dots R_n(t)) \quad (4)$$

Configuração em paralelo

No caso de componentes em paralelo devemos saber qual é a mínima quantidade de componentes que devem estar funcionando para garantir que o sistema funcione. A representação do diagrama de blocos está ilustrada na Figura 2.

No caso do sistema precisar somente de um componente para funcionar, a confiabilidade é estimada da seguinte forma,

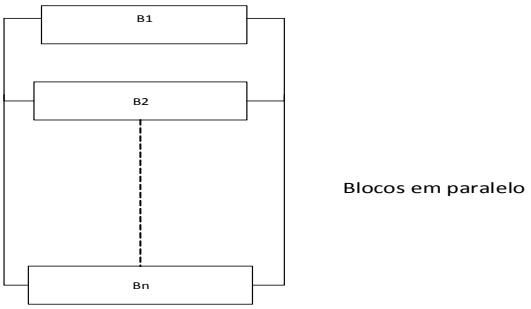

Figura 2 – Blocos em paralelo

$$R_s(t) = 1 - \prod_{i=1}^{m}(1 - R_i(t)) \geq \max(R_1(t), R_2(t), \dots, R_m(t)) \quad (5)$$

O método RAM é uma abordagem estratégica para integrar a confiabilidade, a disponibilidade e a mantenabilidade, usando métodos e técnicas de engenharia para identificar e quantificar falhas em equipamentos e sistemas, com o objetivo de garantir a produtividade do sistema.

Para lidar com está complexidade, metodologias computacionais têm sido adaptadas para simular a resposta do sistema em condições de funcionamento e a falha de seus componentes durante um tempo de missão.

O método de Simulação de Monte Carlo (MCS) é um dos mais usados para reproduzir o comportamento estocástico das mudanças de estado (falha e funcionamento, neste trabalho) dos componentes do sistema e como estes afetam o sistema.

Gerador uniforme de números aleatórios

De todas as distribuições, a distribuição uniforme no intervalo [0,1) denotada por U[0,1) tem um papel muito importante, pois esta distribuição permite gerar números aleatórios de uma forma mais simples que

qualquer outra distribuição.

A função de distribuição acumulada (cdf) equação (1) e a função densidade de probabilidade (pdf) da distribuição U[0,1) equação (2) são:

$$U_R(r) = \begin{cases} 0 & for \quad r < 0 \\ r & for \quad 0 \le r \le 1 \\ 1 & for \quad r > 1 \end{cases} \tag{6}$$

$$u_R(r) = \begin{cases} 0 & for \quad r < 0 \\ 1 & for \quad 0 \le r \le 1 \\ 0 & for \quad r > 1 \end{cases} \tag{7}$$

Gerar números aleatórios pelo método da transformada inversa para distribuições contínuas

$$Dada\ uma\ v.a.\ X \in (-\infty, \infty)\ com\ cdf\ F_X(x)\ e\ pdf\ f_X(x),$$
$$F_X(x) = \int_{-\infty}^{x} f_X(x')dx' = P(X \le x) \tag{8}$$

Dado que $F_X(x)$ é uma função não decrescente, para algum y ∈ [0,1) sua inversa pode ser definida como:

$$F_X^{-1}(y) = \inf\{x: F_X(x) \ge y\} \tag{9}$$

Com esta definição, podemos tomar em conta a possibilidade de que em algum intervalo $[x_s, x_d]$ $F_X(x)$ seja constante (e $f_X(x) = 0$), isto é:

$$F_X(x) = \gamma \qquad for \qquad x_s < x \le x_d \tag{10}$$

Vamos ver que sempre é possível obter valores $X \sim F_X(x)$ a partir de valores R gerados de uma distribuição uniforme $U_R[0,1)$. De fato, se R é uniformemente distribuída in [0,1), tem-se:

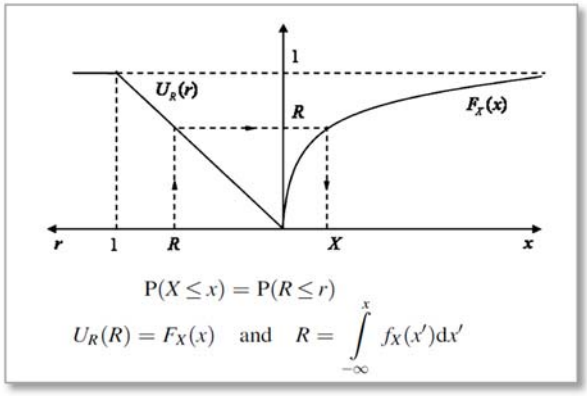

Figura 3 – Método da transformada inversa para gerar números aleatórios.

$$P(R \le r) = U_R(r) = r \tag{11}$$

Correspondente ao número R extraído de $U_R(r)$, calcula-se o número $X = F_X^{-1}(R)$, pode-se vê-lo na Figura 3, para a variável X temos:

$$P(X \le x) = P(F_X^{-1}(R) \le x) \tag{12}$$

Porque F_X é uma função crescente, aplicando isto ao lado direito da equação (3.11) tem-se:

$$P(X \leq x) = P\big(R \leq F_X(x)\big) = F_X(x) \tag{13}$$

Segue-se então que $X = F_X^{-1}(R)$ é extraído de $F_X(x)$. Além disso, $F_X(x) = r$

$$P(X \leq x) = P(R \leq r) \tag{14}$$

em termos da cdf

$$U_R(R) = F_X(x) \quad e \quad R = \int_{-\infty}^{x} f_X(x')dx' \tag{15}$$

Está é a relação fundamental do método da transformada inversa o qual toma algum valor R gerado por uma distribuição uniforme $U_R[0,1)$ e retorna o correspondente valor X gerado pela distribuição $F_X(x)$. Porém, frequentemente ocorre que a cdf $F_X(x)$ não é inversível analiticamente, então não é possível encontrar $X \sim F_X(x)$ como uma função de R~U[0,1).

Analise de confiabilidade e disponibilidade de sistemas via MCS

Vamos considerar um sistema composto por N_C componentes (bombas, válvulas, compressores, reservatórios, etc). Cada componente pode estar em um, dentre vários estados (neste caso, trabalhando ou em falha). Durante o tempo de missão os componentes podem se mover de um estado para outro por uma transição que ocorre de forma aleatória.

Na prática, a análise de confiabilidade de sistemas por meio do método de MCS corresponde à geração virtual de um grande número de experimentos de sistemas com o mesmo comportamento estocástico, cada experimento vai ter um comportamento diferente devido à natureza estocástica do sistema.

São rodadas uma quantidade M de testes, para estimar a confiabilidade do sistema $\hat{R}(t)$ é utilizada a frequência de falhas, e para isso é usado um contador de falhas $C^R(t)$ o qual armazena o número de vezes em que o sistema falhou. Dividindo-se este contador por M, tem-se um estimador de $\hat{F}(t)$, e pela definição de confiabilidade temos

$$\hat{R}_T(t) = 1 - \hat{F}_T(t) = 1 - \frac{C^R(t)}{M} \tag{16}$$

Utilizando-se a definição de disponibilidade: porcentagem de tempo que o sistema esteve disponível para fazer sua tarefa durante o tempo de missão T_M, pode-se estimar a disponibilidade $\hat{A}_T(t)$. Subtraindo-se do tempo de missão T_M o tempo que o sistema ficou no estado de falha downtime e dividindo-o por T_M, pode-se acumular este valor na variável $DT^R(t)$ para os M testes e finalmente dividi-lo por M para obter o $\hat{A}_T(t)$.

$$Para\ cada\ teste\ DT^R(t) = \frac{T_M - downtime}{T_M} \tag{17}$$

Depois dos M testes

$$o\ estimador\ \hat{A}_T(t) = \frac{DT^R(t)}{M} \tag{18}$$

Algoritmo Genético *GA*

Há várias técnicas eficientes de otimização que foram inspiradas nos mecanismos biológicos [4]. Os

GAs são técnicas de busca e otimização que funcionam de forma aleatória e paralela baseados em um principio natural genético, que pode realizar pesquisas e otimização de cenários complexos. Os GAs são muito eficientes quando os espaços de busca são muito grandes e complexos. Uma importante componente deste processo é a programação genética e estratégias de evolução que podem ser designadas para a otimização de um ou de vários critérios.

Embora, existam alguns problemas da vida real que necessitem de otimização de mais de um critério que frequentemente são contraditórios. Dessa forma, estas necessidades requerem técnicas de Otimização Multiobjetivo (*Multiobjective Optimization MOO*). Nesse contexto, neste trabalho usa-se o Algoritmo Genético Multiobjetivo (MOGA) [6]. Devido ao fato que os processos MOOs envolvem várias funções mérito conflitantes, a solução final de um processo MOO é um subconjunto da Solução Ótima de Pareto **Erro! Fonte de referência não encontrada.**. Os resultados da Solução Ótima de Pareto contêm um número de indivíduos chamados de não dominantes dos quais o usuário pode escolher o mais promissor. Para este trabalho será encontrada a melhor configuração possível para o caso de estudo.

O MOGA selecionado para este trabalho foi o GA de classificação elitista não dominante (*Elitist Non-Dominated Sorting GA*) NSGA-II [7]. O NSGA II foi proposto para eliminar os indivíduos mais fracos da amostra, e em cada rodada do algoritmo é feita uma classificação de todos os indivíduos colocando, colocando os mais eficientes na primeira camada ou na curva frente de Pareto. Estes cromossomos recebem a classificação #1, e esse processo é então repetido para o restante dos indivíduos, com o intuito de se obter as classificações #2...#m, onde a classificação m representa os indivíduos mais fracos da amostra. Dentro de cada classificação é calculada a distância de aglomeração (métrica para medir a densidade da solução).

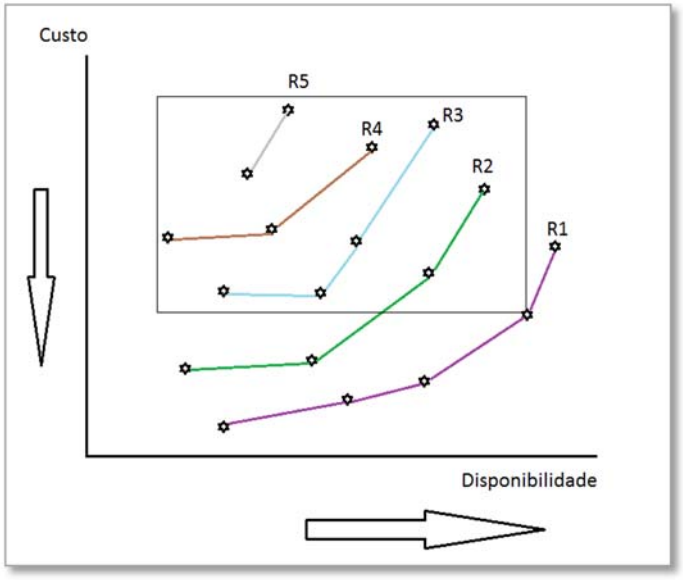

Figura 4 – Classificação de indivíduos maximizando Disponibilidade e minimizando Custo.

Para se implementar o GA elitista, começa-se com uma população de cromossomos ou indivíduos escolhidos de forma aleatória. Assim cada cromossomo será avaliado com base em dois valores requeridos de aptidão (por exemplo: disponibilidade e custo) estimados por meio do MCS. Em seguida, os cromossomos serão classificados e se reproduzirão pelo método de seleção. Durante essa seleção, os cromossomos com melhor aptidão recebem mais chances de serem cruzados com os outros. O método de seleção utilizado pelo NSGA II será a roleta binária. Esta técnica seleciona dois indivíduos da amostra e avalia a classificação de ambos. O cromossomo melhor avaliado será um futuro pai e se ambos tiverem a mesma classificação aquele com menor distância de aglomeração será selecionado, e o outro será dispensado. Este método garante que o individuo mais dominante da amostra atual sempre prevalecerá e o mais fraco sempre será descartado. Se o tamanho da amostra é impar o último sorteio terá três indivíduos e só prevalecerá o melhor deles.

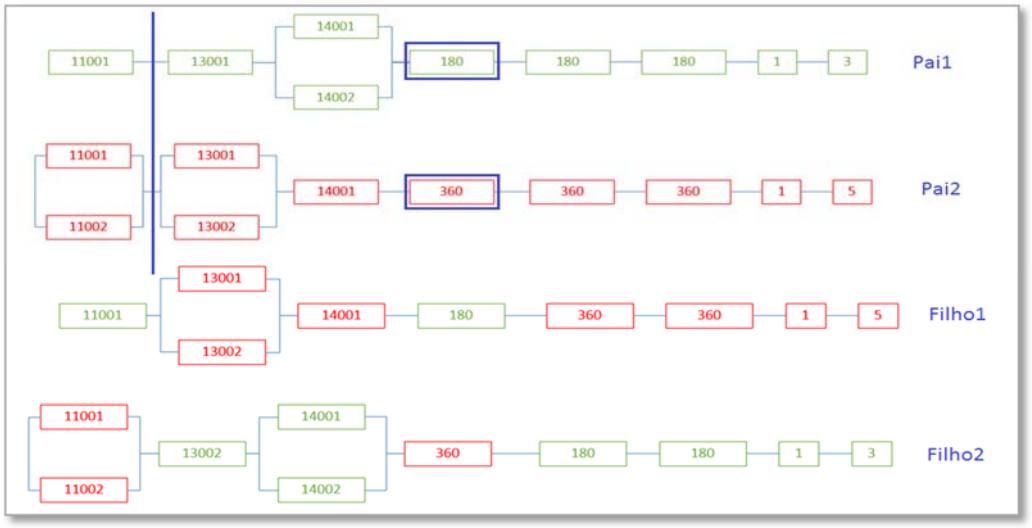

Figura 5 – Método de cruzamento.

A cada dois sorteios da roleta binária é feito um cruzamento que dá origem a dois novos indivíduos para se manter sempre o tamanho da amostra. Além do cruzamento para cada cromossomo também é dada uma chance de mutação para se manter a diversidade. Assim os processos de recombinação ou reprodução por mutação ou cruzamentos aplicadas na seleção dos cromossomos são mecanismos que conservaram informações criticas que promovem uma melhor produção de soluções na próxima geração. Este processo continua até que o critério estabelecido para a rescisão seja cumprido.

Características dos GAs

Os GAs trabalham com uma codificação interna associada às características próprias dos cromossomos. Estas propriedades incluem as variáveis problemáticas, mas os GAs não alteram as variáveis mérito estabelecidas. O conjunto de indivíduos com maior dominância é pesquisado simultaneamente a partir de múltiplos pontos e isto envolve simultaneidade no processo de busca. Por outro lado, os GAs caracterizam-se por ser uma técnica de busca cega por meio de amostragens utilizando somente a informação da aptidão requerida.

Protótipo do software OptimizLabRisco.

Os códigos do OptimizLabRisco foram desenvolvidos com Programação OOP [9], compilados na versão C++ 11 [10], utilizou-se o ambiente de programação Visual Studio 2015 da Microsoft e como padrão de desenho de software utilizado foi a “FABRICA DE SOFTWARE” [10]. Porque estas ferramentas? A continuação um curto detalhamento de alguns bons atributos delas.

Programação Orientada a Objetos (OOP).

Os programas escritos no paradigma OOP possuem mais vantagens que desvantagens com respeito à Programação Estruturada. Algumas vantagens que foram consideradas neste projeto:

• A OOP é a metodologia de programação mais utilizada nos últimos tempos. Esta propriedade é evidenciada na quantidade de linguagens de programação de alto nível que usam dito paradigma entre eles: Java, C#, Python, Ruby, C++.

• A OOP permite desenvolver independentemente e/ou paralelamente várias partes do pacote final do software. Além disso, o código desenvolvido em OOP é altamente reutilizável, o que da muita flexibilidade ao software, pois permite uma expansão do projeto em complexidade criando mais classes e/o novas interfaces.

• A manutenção de um código em PPO é menos complicada, já que se pode adicionar e/ou eliminar atributos e funções das classes sem afeitar consideravelmente outras classes.

• A OOP é uma metodologia de programação na qual a representação do sistema está mais perto do que é no mundo real, uma vez que podemos descrever as características e as funções dos subsistemas.

A linguagem de programação C++ e uma linguagem de Programação de alto nível Orientada a Objetos e baseada em c, essa combinação permite utilizar as capacidades de c em um estilo Orientado a Objetos (citação).

Caso de estudo

O caso de estudo é constituído por um subsistema de um complexo sistema de tratamento de águas. O subsistema possui três blocos de componentes da Figura 7 (tanques de água, válvulas e bombas centrífugas), cada bloco pode ter componentes em redundância.

Figura 6 – Sistema de resfriamento de água

Figura 7 – representação Sistema de resfriamento de água

O sistema opera 24 horas por dia e 7 dias por semana. O objetivo é determinar qual é a melhor configuração dentro de todas as possíveis acatando todas as restrições. As restrições estão determinadas pela disponibilidade de equipamentos, a quantidade máxima de redundâncias permitidas para cada bloco de componentes, o mínimo de componentes para manter um bloco funcionando, a quantidade mínima e máxima de equipes de manutenção e os intervalos mínimos e máximos entre manutenções preventivas para cada bloco.

As restrições do sistema são:
• Cada bloco de componentes precisa de um componente para funcionar.
• Cada bloco pode ter máximo um componente em redundância.
• Se um componente falar só um equipe de manutenção pode ser alocado em aquele componente.
• Pela restrição anterior a quantidade de equipes não deve ser maior que a quantidade total de componentes do sistema.

Tabela 1 – Parâmetros dos componentes do estudo de caso.

Ref.	Equip.	Custo	Distr. Falha	Parâmetro #1	Parâmetro #2	Distr. Reparo	Parâmetro
11001	Tanque	25	Exp.	4.32	---	Exp.	172.8
11002	Tanque	25	Exp.	3.32	---	Exp.	87.8
13001	Válvula	150	Exp.	4.38	---	Exp.	175.2
13002	Válvula	205	Weibull	1.2	0.5	Exp.	131.4
14001	Bomba	90	Weibull	2.0	0.3	Exp.	175.2
14002	Bomba	120	Weibull	2.4	0.25	Exp.	87.6

Nota: As unidades dos parâmetros estão em anos, por exemplo, para o Tanque com referência 11001 o MTTF = 360/4.32 = 83.333 dias e o MTTR = 360/172.8 = 2.08 dias.

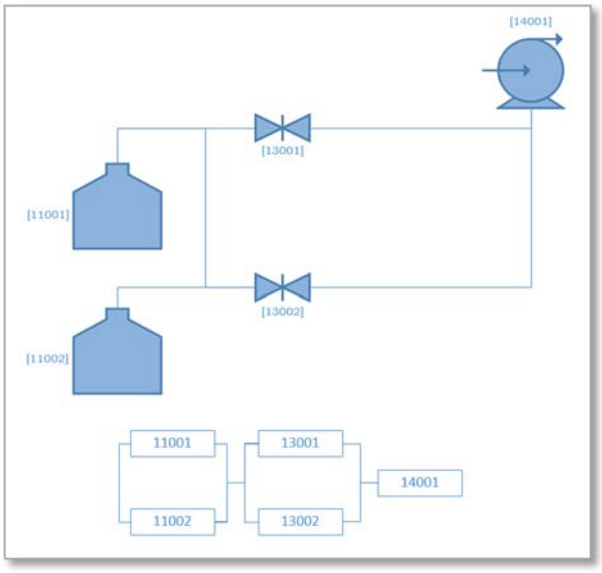

Figura 8 – Representação em diagrama de blocos do sistema.

O principal elemento do GA é o cromossomo, nele se deve armazenar a informação relacionada com as características (Gene) que determinam a aptidão do individuo. Como fio dito antes, Neste projeto foi utilizado como metodologia de programação a OOP, seguindo está metodologia definiu-se a classe "Chromosome" para representar um sistema. Um componente muito importante do cromossomo é o Gen. O Gen é um segmento de ADN, neste contexto se definiu a classe "Gen" para representar um bloco. As moléculas do ADN contêm as instruções genéticas que determinam as propriedades dos indivíduos, continuando com a analogia biológica determinou-se que a classe "Molecule" representa um componente, pois este último contem toda a informação (por exemplo: taxas de falha e reparo, quantidade de possíveis estados, tipos de distribuições para cada estado, custos associados etc.) para poder estimar por meio da MCS as variáveis mérito segundo a forma como estejam configurados os componentes no sistema.

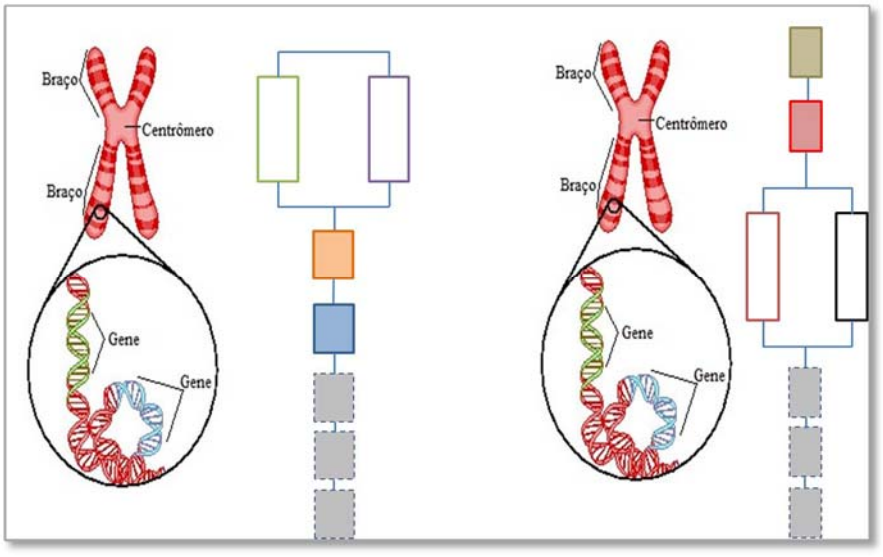

Figura 9 – Representação do sistema como um cromossomo.

Outro ponto importante é que o Gene não tem um tamanho fixo. Utilizando está propriedade foi possível colocar dados (não objetos) como se fossem segmentos de ADN de tamanho um sem danificar a estrutura do cromossomo, estes dados representam características muito importantes do sistema tais como quantidade de equipes de manutenção e se o sistema terá ou não manutenção preventiva (para o caso de querer avaliar a confiabilidade do sistema).

Resultados

Primeiro será gerada toda a população respeitando as restrições antes mencionadas, neste caso se geraram 602 indivíduos Figura 10. Cada amostra e de 20 indivíduos. O objetivo é minimizar os custos e maximizar a disponibilidade do sistema.

Para a simulação de cada sistema (indivíduo) o tempo de missão será de 10 anos e foram feitas 1000 simulações para cada rodada.

A primeira amostra é gerada de forma aleatória e ao final da quarta geração pode-se ver o algoritmo atingiu a curva frente de Pareto.

O algoritmo entrega a quarta geração como sendo um subconjunto da população onde os usuários poderão focar seus esforços em analisar dito conjunto de configurações do sistema para escolher a melhor configuração. (tabela xx)

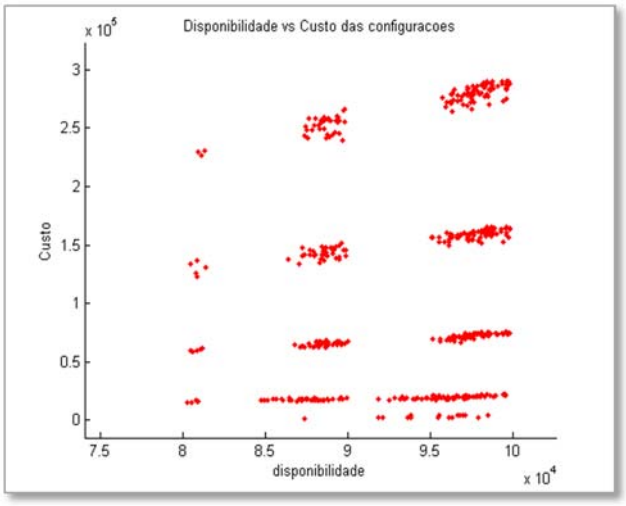

Figura 10 – População total.

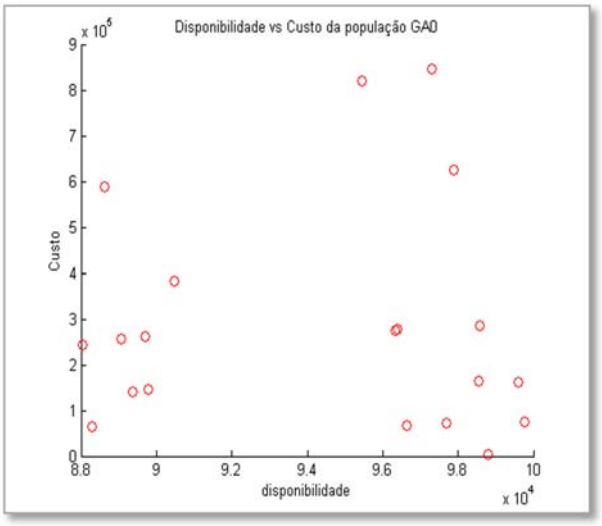

Figura 11 – Geração inicial

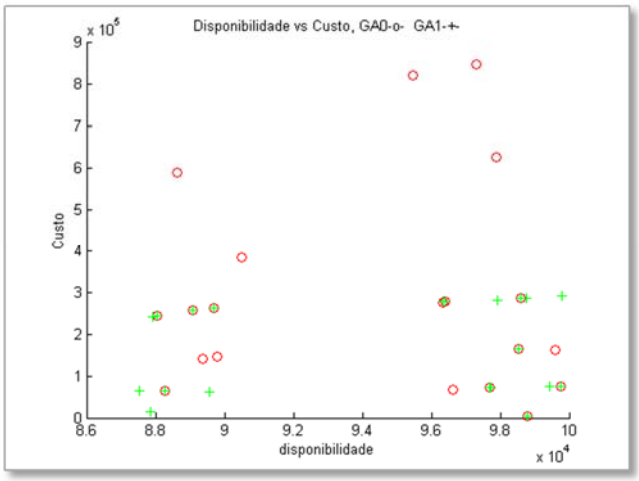

Figura 12 – Primeira Geração vs. Geração inicial

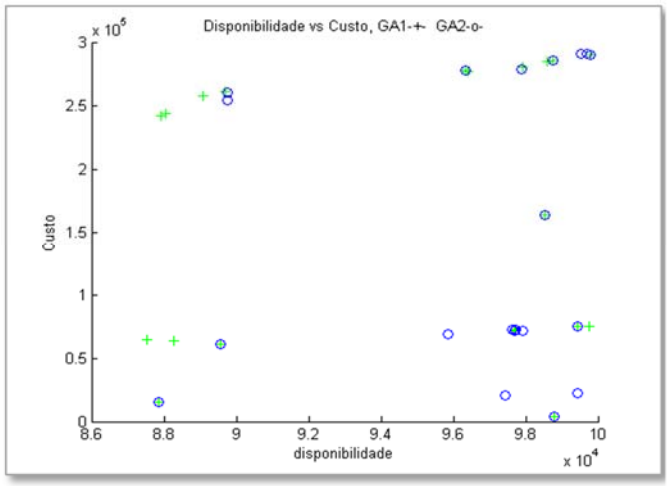

Figura 13 - Segunda Geração vs. Primeira Geração.

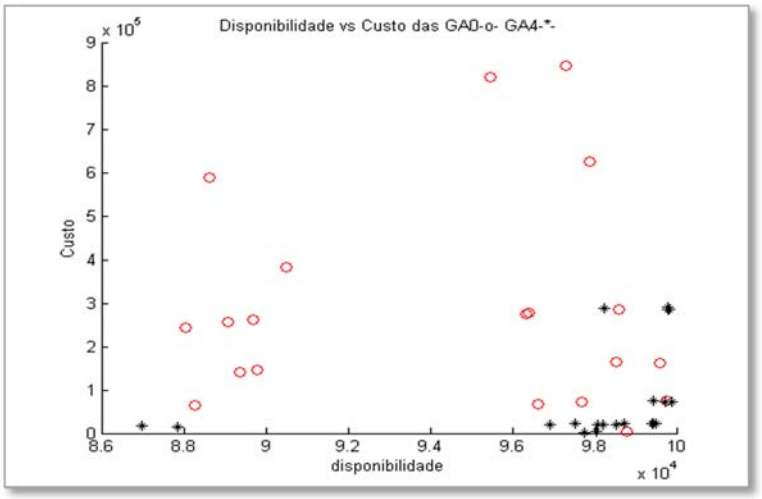

Figura 14 – Quarta Geração vs. Geração inicial.

Tabela 2 – Geração Inicial.

	Dispon.	Custo	Bloco 1	Bloco 1	Bloco 2	Bloco 2	Bloco 3	Bloco 3	Equi. Manut.
1	98800	4363	11002	11001	13002	13002	14002	14001	1
2	97302,8	845642	11001	11001	13002	13002	14002	---	7
3	88283,3	63421	11001	13002	14002	---	---	---	2
4	89075	257513	11001	13002	13002	14001	---	---	4
5	89700	260950	11001	13002	13002	14002	14001	---	4
6	96388,9	277010	11002	13002	14001	14001	---	---	4
7	89369,5	139632	11001	13002	14001	14001	---	---	3
8	99588,9	162734	11001	11002	13002	13002	14001	14001	3
9	96644,4	67360	11002	13002	14001	14002	---	---	2
10	90486,1	384092	11001	13002	13002	14002	14002	---	5
11	96347,2	273711	11002	13002	14001	---	---	---	4
12	98586,1	284790	11001	11002	13002	13002	14001	---	4
13	98544,4	163742	11001	11002	13002	13002	14002	---	4
14	88036,1	243070	11001	13002	14002	14002	---	---	4
15	88625	588829	11001	13002	14001	---	---	---	6
16	97700	71182	11002	11002	13002	14002	14002	---	2
17	97886,1	626096	11001	11002	13002	14001	14002	---	6
18	95463,9	818758	11002	13002	14001	---	---	---	7
19	89797,2	145202	11001	13002	13002	14002	14002	---	3
20	99755,6	75068	11002	11002	13002	13002	14001	14002	2

Tabela 3 – Quarta Geração.

	Dispon.	Custo	Bloco 1	Bloco 1	Bloco 2	Bloco 2	Bloco 3	Bloco 3	Equi. Manut.
1	98716.7	21761	11002	11001	13002	13002	14002	14001	1
2	97766.7	2580	11002	11002	13002	---	14001	14002	1
3	98044.4	4390	11002	11001	13002	13002	14002	14001	1
4	98094.4	20093	11002	11002	13002	---	14002	14002	1
5	96922.2	20120	11001	11002	13002	13002	14001	---	1
6	99447.2	74839	11002	11001	13002	13002	14002	14001	2
7	98527.8	20088	11001	11002	13002	13002	14001	---	2
8	97547.2	21368	11002	11001	13002	13002	14002	14001	1
9	99436.1	22190	11002	11002	13002	13002	14001	14002	1
10	87863.9	15386	11001	---	13002	---	14002	---	1
11	99402.8	22021	11002	11002	13002	13002	14001	14002	1
12	86997.2	16163	11001	---	13002	---	14002	---	1
13	99800	289767	11002	11002	13002	13002	14001	14002	4
14	98213.9	19804	11002	11002	13002	---	14002	14002	1
15	99497.2	21213	11002	11002	13002	13002	14001	14002	1
16	98236.1	289203	11002	11002	13002	---	14002	14002	4
17	99811.1	286566	11002	11002	13002	13002	14001	14002	4
18	99877.8	73404	11002	11002	13002	13002	14001	14002	2
19	99719.4	72590	11002	11002	13002	13002	14001	14002	2
20	99777.8	285768	11002	11002	13002	13002	14001	14002	4

CONCLUSÕES E COMENTÁRIOS FINAIS

O método mostrou ser eficiente não só na identificação da fronteira de Pareto (tendo utilizado 4 gerações), como também no tempo de execução, pois para os pais selecionados na roleta binária não é preciso estimar de novamente as variáveis mérito; nestas condições a quantidade de vezes em que se faz necessário a execução do método de MCS é muito pequena comparada com a quantidade de vezes requerida para avaliar a população total, composta por todas as possíveis configurações para o sistema sob análise.

Como as funções mérito das colunas dois e três da Tabela 3 são estimadas utilizando-se o método de MCS, vários valores diferentes (mas muito similares) podem representar a mesma configuração de sistema, como são os casos das linhas: a) linhas 1 e 3; b) linhas 4 e 14; c) linhas 18 e 19; d) linhas 17 e 20, e; e) linhas 13 e 17. Porém, uma configuração foi gerada três vezes (linhas 9, 11 e 15). Este fenômeno acontece dentro do GA sugerindo que eles possam ser os melhor avaliados, constituindo-se uma forma de convergência. Neste contexto, a configuração da linha 9 (configurações 9, 11 e 15 são representadas pelo mesmo cromossomo) poderia ser considerada a mais promissora.

Visual Studio e a utilização dos ponteiros de C++ permitiram entender melhor a complexidade da estrutura de dados utilizada para o método de classificação das soluções não dominantes utilizado no algoritmo NSGA_II.

AGRADECIMENTOS

Os autores agradecem o apoio financeiro do Programa de Desenvolvimento de Recursos Humanos (PRH19) da Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP) e da Petrobrás, através do seu programa de bolsas.

REFERÊNCIAS

[1] ZIO, E.. *The Monte Carlo Simulation Method for System Reliability and Risk Analysis.* Springer-Verlag London. (2013).

[2] BLUM, C.; ROLI, A.. *Metaheuristics in combinatorial optimization*: Overview and conceptual comparison. ACM Computing Surveys 35(3) 268–308. (2003).

[3] HOLLAND, J. H.. *Adaptation in Natural and Artificial Systems*. An introductory Analysis with Applications to Biology, control and Artificial Intelligence. MIT Press. (1992).

[4] MODARRES, M. KAMINISKIY, M. AND KRIVTSOV, V.. *Reliability engineering and risk analysis*: a practical guide. 2nd edition. (2010).

[5] LINDEN, R.. *Algoritmos Genéticos*. Ciência Moderna, Rio de Janeiro. (2006).

[6] DAVIS, L. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, (1990).

[7] HORN, J.; NAFPLIOTIS, N.; Goldberg, D. E.. *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*. In Proceedings of the 1st IEEE conference of Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, NJ, USA, 27-29 June 1994; Volume 1, pp. 82-87.

[8] DEB, K., PRATAP, A., AGRAWAL, S. AND MEYARIVAN, T.. *A fast and Elitist Multiobjective Genetic Algorithm:* NSGA-II. IEEE Transactions on Evolutionary Computation. 6:182-197. (2002).

[9] SOMMERVILLE, I.. *Ingenieria del Software*. 7th Edtion Spanish – Pearson Education. (2005).

[10] DEITEL, P. AND DEITEL, H.. *How to Programming C++*, Eighth edition, Prentice Hall. (2012).

[11] FREEMAN, E.. Head First Design Patterns. O´Reilly. (2004).