UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

GUSTAVO ALVES PACHECO

RESOLUÇÃO DE UM PROBLEMA DE SCHEDULING

Uberlândia-MG

2019

GUSTAVO ALVES PACHECO

RESOLUÇÃO DE UM PROBLEMA DE SCHEDULING

Trabalho apresentado ao curso de Engenharia de Computação da Universidade Federal de Uberlândia, como requisito parcial de avaliação da disciplina de Algoritmos Genéticos.

Prof.: Keiji Yamanaka

Uberlândia-MG

2019

SUMÁRIO

1. INTRODUÇÃO

Anteriormente, havia sido implementado um algoritmo genético para maximizações de funções de várias variáveis, com ajustes de todos os parâmetros, incluindo a própria função a ser maximizada. O desafio atual é resolver um problema de *scheduling*.

Tal problema se baseia no agendamento da manutenção de diferentes geradores em uma usina, de forma a manter um nível constante de potência líquida. Todas as máquinas deverão passar uma vez pelo processo. Algumas delas precisam ficar mais de um período paradas, sendo que, ao longo do ano, existem quatro intervalos disponíveis.

Cada um desses intervalos possuem uma potência máxima de demanda, que sempre deve ser cumprida. Quando uma máquina entra em manutenção, deve ficar parada por intervalos consecutivos e no mesmo ano. Deve-se também levar em consideração o fato de que cada gerador tem uma capacidade diferente.

Tendo em mente todos esses aspectos, novas funcionalidades precisam ser adicionadas ao algoritmo já feito. A primeira está no novo design dos cromossomos, que devem ser capazes de representar as diferentes possibilidades para cada período, ao mesmo tempo que permitem passar pelas operações genéticas já implementadas.

Outra problemática que surge é a formulação da função a ser minimizada/maximizada e a forma como o fitness será trabalhado. Além disso, novas técnicas para visualização dos resultados deverão ser desenvolvidas. Não se faz necessário reescrever todo o código, como feito anteriormente, sendo preciso apenas algumas modificações pontuais.

A reserva líquida P_L do sistema é dada por:

$$P_L = P_T - P_P - P_D$$

Sendo P_T a Potência total instalada, P_P a potência perdida por parada de máquina e P_D a potência máxima de demanda.

A tabela 1 abaixo mostra as informações de cada unidade de geração de energia, enquanto a 2 mostra a potência máxima de demanda a cada período.

Tabela 1: Informações sobre cada unidade geradora

Número da unidade	Capacidade (MW)	Número de intervalos de manutenção / ano
1	20	2
2	15	2
3	35	1
4	40	1
5	15	1
6	15	1
7	10	1

Tabela 2: Potência máxima de demanda a cada período

Período	Potência máxima de demanda (MW)
1	80
2	90
3	65
4	70

2. OBJETIVOS

- Modificar o algoritmo genético implementado até o momento para que resolva um problema de agendamento de manutenções
- Desenvolver novo design para os cromossomos
- Formular a equação a ser trabalhada
- Implementar condicionais de população e indivíduo
- Desenvolver forma de visualização dos resultados

3. MATERIAIS E MÉTODOS

Os métodos e materiais utilizados foram iguais aos usados anteriormente, tais elementos estão citados abaixo.

A linguagem utilizada foi o Racket, uma linguagem de uso geral, baseada em Scheme e Lisp. Como interface de desenvolvimento, foi utilizado o DrRacket.

Para confecção da interface gráfica, utilizou-se a linguagem *racket/gui*, um recurso nativo da própria linguagem Racket. O processo de plotagem foi realizado através das funções do pacote *plot*.

Como métodos, utilizou-se uma abordagem *top-down*, com auxílio de um framework SCRUM para desenvolvimento ágil. Em algumas situações, uma visão *bottom-up* foi adicionada ao projeto, na parte de incremento da própria linguagem, ao criar novos operadores e funcionalidades. O código produzido segue majoritariamente

uma abordagem funcional, mas possui elementos procedurais, para facilitar a leitura de algumas funções.

Durante o desenvolvimento, a documentação de funções se mostrou fundamental. Principalmente, para que algumas estruturas de dados fossem melhor especificadas, e a execução do código fosse correta.

4. RESULTADOS E DISCUSSÕES

Inicialmente, foi feito o design do cromossomo. Era necessária alguma forma de manter um registro sobre quais máquinas estavam operando e quais estavam em manutenção em cada período. Em primeira instância, pensou-se em utilizar um cromossomo com 35 bits, sendo que cada 5 bits representava uma máquina. Dentro desses 5 bits, os 3 primeiros correspondiam ao número da máquina, e os dois últimos ao período no qual a manutenção se iniciava.

Essa ideia logo foi descartada, já que uma máquina poderia aparecer duplicada dentro do cromossomo, e alguma outra poderia não ser representada. Com isso, bolou-se um sistema com 28 bits, em teoria, sendo que cada 7 representava o estado do período. Ou seja, cada máquina era exibida como um bit a cada período, e cada período representado como um conjunto de 7 bits.

Caso o bit esteja como 1, a máquina está em manutenção. Com isso, era possível calcular a potência a cada período, sabendo quais máquinas estavam ativas. E o problema de duplicidade já não poderia mais acontecer, já que cada período representava um espaço fixo dentro do cromossomo. Na prática, cada indivíduo foi representado como uma lista de quatro elementos, sendo cada elemento uma *string* de 7 caracteres (bits).

Em seguida, a função a ser trabalhada foi desenvolvida. Notou-se que ao longo do ano a potência líquida absoluta sempre continuava com o mesmo valor, independente da ordem das manutenções, já que cada máquina parava apenas uma vez e, no contexto geral, sempre representam as mesmas perdas. Esse valor de potência líquida era fixo em 110 MW. Por causa disso, não era possível maximizar a potência total ao longo do ano. Também não era vantajoso trabalhar com a maximização de cada período, pois os requisitos de demanda e manutenções em mais de um intervalo acabariam por deixar a potência bem desnivelada.

Para solucionar esse problema, optou-se por minimizar o desvio padrão das potências líquidas de cada período. O desvio padrão representa o quanto um conjunto de dados é uniforme. Tal propriedade é exatamente o que se desejava e, portanto, foi adotada como medida de aptidão. Claro que foi necessário multiplicar a função de desvio padrão por -1 e somar com algum valor constante para que todos os dados fossem positivos.

A função a ser maximizada foi a seguinte:

$$f = 500 - \sqrt{\frac{\sum_{i=1}^{4}(P_i - P_M)^2}{3}}$$

Tendo em mãos a representação do indivíduo e a função a ser maximizada, era possível começar a pensar na geração das populações. Nesse ponto, deve-se lembrar dos condicionais impostos. O que foi feito é que, a cada novo indivíduo gerado (seja aleatoriamente, ao inicio da simulação, ou pelos processos genéticos), o período é validado conforme a potência máxima de demanda, e o indivíduo em si é analisado, para garantir que cada máquina cumpra as manutenções sequenciais e não fragmentadas.

Esta validação acaba por deixar a execução mais lenta, mas da forma como foi implementada, utilizando recursões de cauda aninhadas (que são compiladas em iterações extremamente eficientes), o processo ficou o mais rápido possível. A partir daí, as pendências foram a respeito da adequação dos operadores genéticos de crossover e mutação, bem como ajuste na forma de seleção (torneio, seleção) e melhorias no processo de elitismo.

Com o algoritmo genético funcionando, partiu-se para o processo de visualização dos dados, para garantir que a execução do código estivesse correta. O gráfico de performance foi mantido, levando em consideração que o processo atual é de minimização, e a aptidão de cada indivíduo é o desvio padrão das potências líquidas de cada período. Um gráfico de animação foi adicionado, representando os indivíduos a cada população. Esses indivíduos são linhas indicando as potências por período. Quanto mais constante a linha, melhor a aptidão do indivíduo.

Além dos gráficos, uma tabela foi adicionada, para melhor análise dos resultados finais (melhor indivíduo). Por último, vários testes foram feitos, variando os valores das probabilidades de mutação e crossover, o valor do elitismo, a forma de seleção, o número de indivíduos e o número de gerações. Alguns testes realizados estão listados abaixo. O melhor resultado possuía aptidão de 2,88675, ou seja, um desvio padrão muito próximo de 0, como queríamos obter. Esse resultado foi encontrado em várias execuções, com diversos ajustes diferentes. Algumas configurações diferentes de *scheduling* podem possuir o mesmo desvio, já que nas informações fornecidas para cada máquina existem unidades gêmeas.

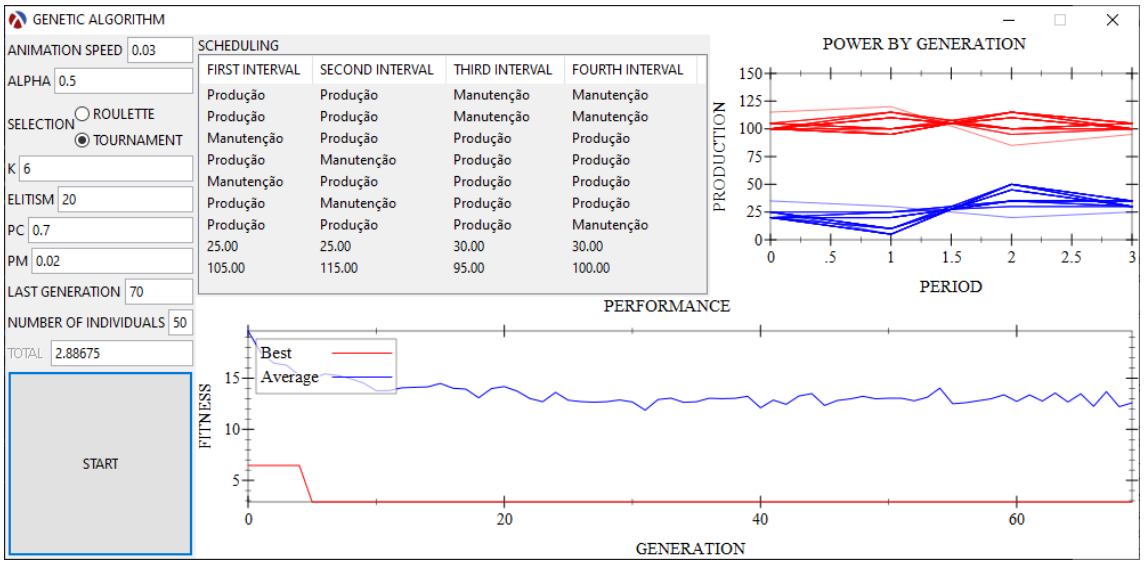

Figura 1: Teste utilizando torneio e elitismo altos. Nota-se que a média ficou relativamente constante, e é possível ver a evolução do melhor indivíduo.

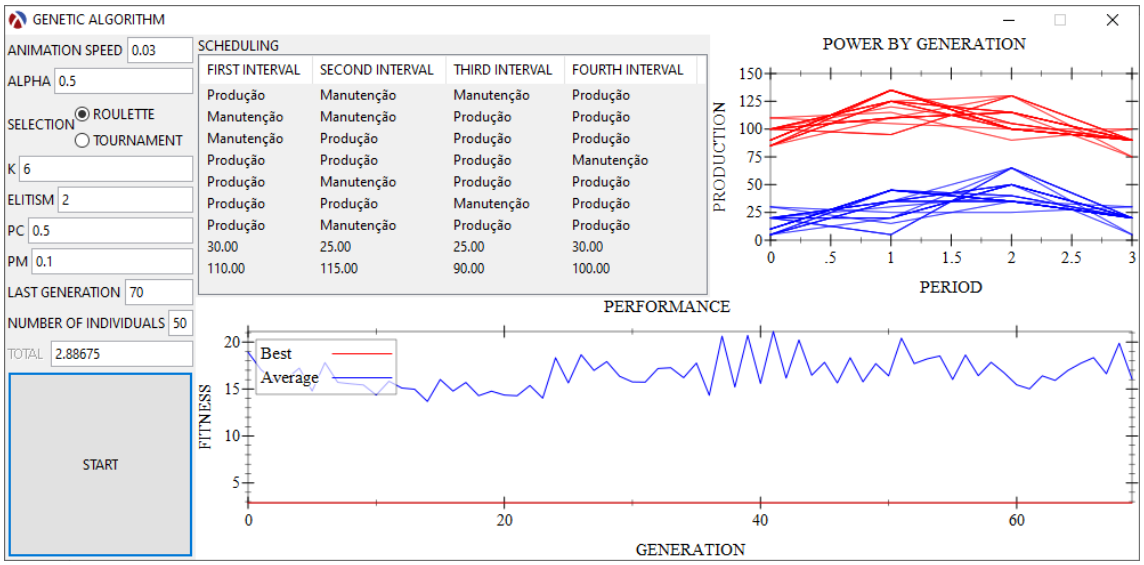

Figura 2: Teste utilizando roleta. Percebe-se que, apesar de ser encontrado o melhor indivíduo, semelhante ao teste 1, a ordem das manutenções não é a mesma.

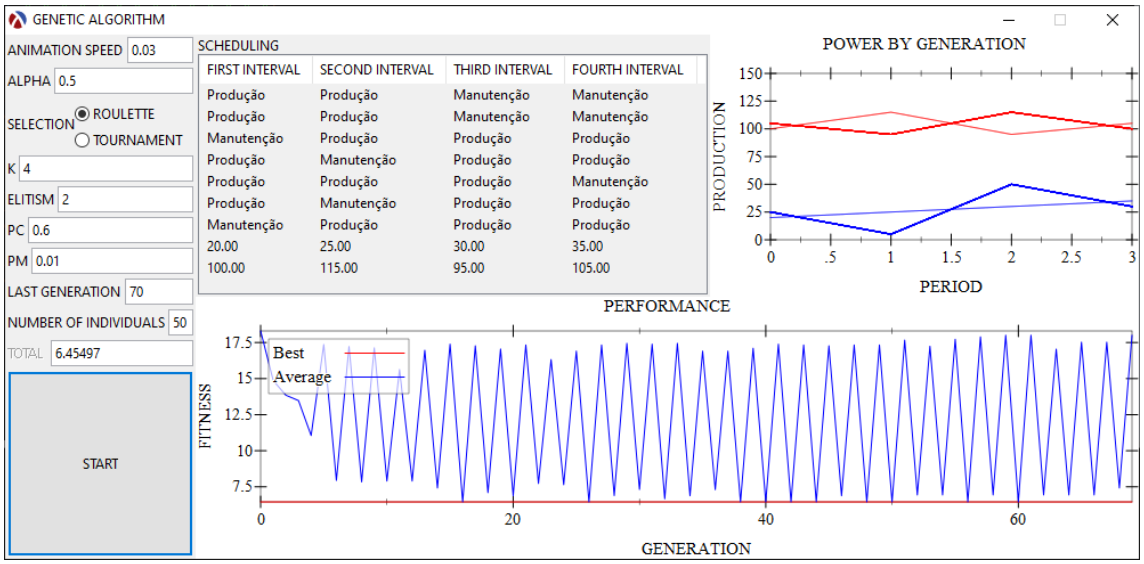

Figura 3: Grande variação no valor da média, ao usar roleta. Causada pela presença dos condicionais, que limitam os valores de aptidão a resultados discretos, e não contínuos, como anteriormente.

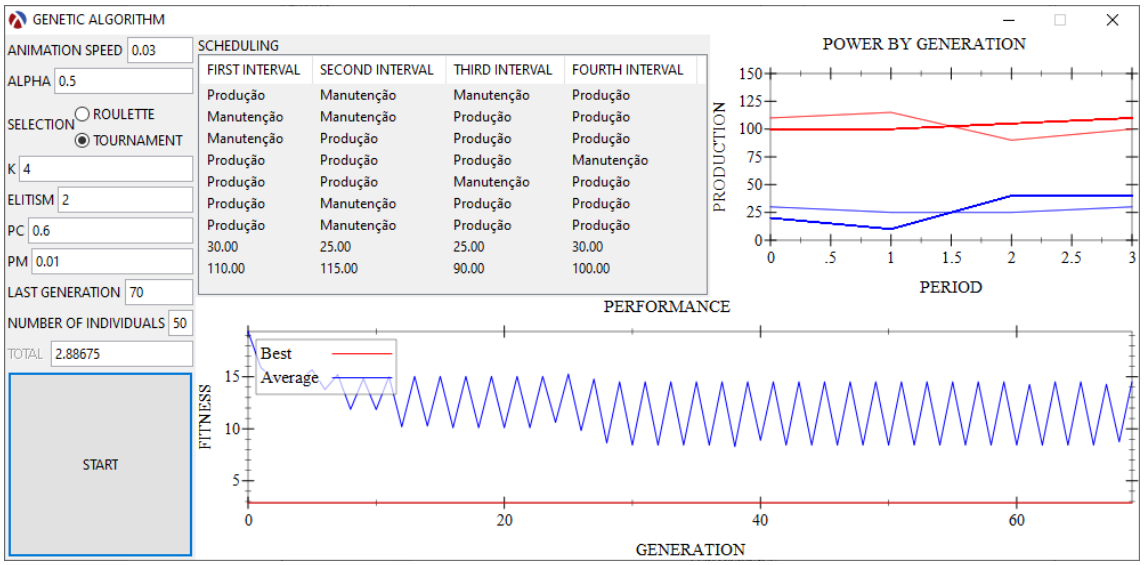

Figura 4: Variação na média, reduzida em comparação com a figura 3, devido ao método de seleção por torneio.

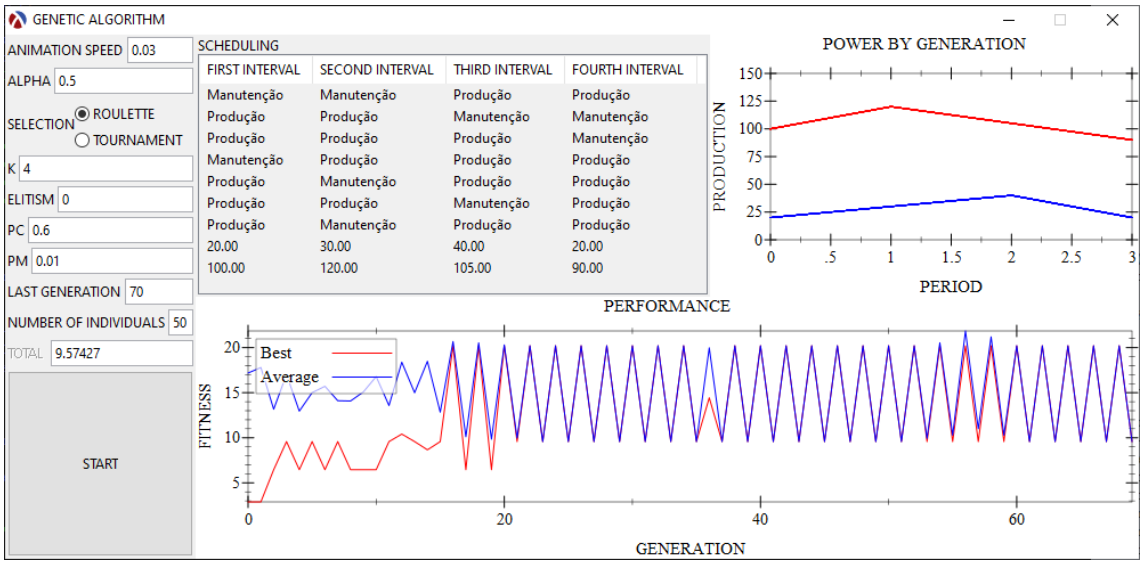

Figura 5: Péssimo resultado obtido ao eliminar a presença do elitismo. Como dito anteriormente, dados discretos geram oscilações nos valores da média e, nesse caso, no melhor indivíduo.

5. CONCLUSÃO

A nova abordagem utilizada, de resolver problemas reais utilizando algoritmos genéticos, se mostra bem interessante. Em questão de poucos segundos de execução do código, todo o planejamento é feito, garantindo o melhor gerenciamento de recursos, da forma como o usuário desejar.

É impressionante observar a evolução do espectro de aplicações que surgem a cada incremento no código. Cada vez mais, novas possibilidades de aplicações dessa ferramenta surgem. Percebe-se que valores discretos, como os desse experimento, proporcionam convergência rápida, mas uma média geral da população bem ruim. Nota-se que o elitismo é fundamental para alcançar algum resultado palpável, e que o torneio novamente se mostra um método de seleção mais eficiente que a roleta, para as configurações mais comuns.

Como mostrado nos testes, várias formas de agendamento podem ser utilizadas, que produzem o mesmo resultado final. Essas formas foram encontradas com certa facilidade, e com uma precisão considerável. A confiabilidade do algoritmo, para esse caso, se mostrou alta.

6. REFERÊNCIAS BIBLIOGRÁFICAS

Mallawaarachchi, V. (7 de Julho de 2017). *Introduction to Genetic Algorithms*. Fonte: Towards Data Science: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

Montesanti, J. d. (s.d.). *Seleção natural*. Fonte: InfoEscola: https://www.infoescola.com/evolucao/selecao-natural/

Tomassini, M. (s.d.). A Survey of Genetic Algorithms. *Annual Reviews of Computational Physics, Volume III*.

Yamanaka, P. K. (Agosto de 2017). *ALGORITMOS GENÉTICOS: Fundamentos e Aplicações*.

Zini, É. d., Neto, A. B., & Garbelini, E. (18 de Novembro de 2014). ALGORITMO MULTIOBJETIVO PARA OTIMIZAÇÃO DE PROBLEMAS RESTRITOS APLICADOS A INDÚSTRIA. *Congresso Nacional de Matemática Aplicada à Indústria*.