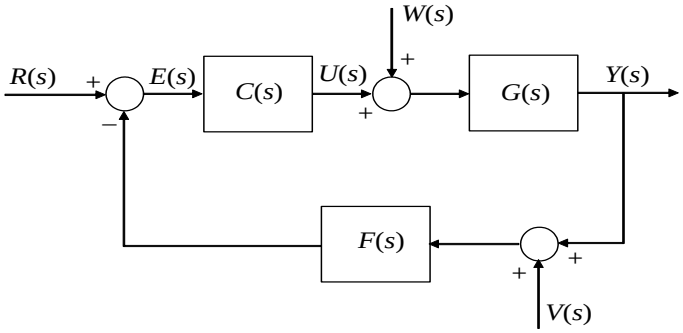
EA072 – Turma A (2s2019)
Exercícios de Fixação de Conceitos – EFC 2
Atividade Individual – Peso 6 – Data de Entrega: 28/11/2019

Questão 1) Controle PID empregando um algoritmo evolutivo, que opera como uma meta-heurística de otimização em espaços contínuos.

Nos laboratórios de controle de processos da FEEC/Unicamp, como EA619 e EA722, estudam-se projetos de controle PID (controle proporcional-integral-derivativo em malha fechada), onde é necessário definir os valores de k_P, k_D e k_I para se maximizar um determinado conjunto de critérios de desempenho. No controle PID, a ação de controle vai ser proporcional ao erro de saída da planta, à derivada deste mesmo erro e à integral deste mesmo erro. São vários critérios de desempenho que podem ser considerados no projeto (individualmente ou conjuntamente), como tempo de subida, tempo de acomodação, sobressinal, erro de regime, margem de ganho e margem de fase. Não é simples definir simultaneamente os valores dos ganhos k_P, k_D e k_I, pois há interferência entre esses ganhos. Mesmo assim, há formas sistemáticas de projeto, que são estudadas em EA721 (curso teórico associado ao laboratório) e em EA722. Nesta atividade, iremos supor que o projetista não domina conceitos de projeto de controle PID, mas tem a tarefa de projetar um controlador PID visando atender três critérios de desempenho. Para tanto, o projetista está convidado a empregar uma versão de algoritmo evolutivo em Matlab, com operadores genéticos dedicados à otimização em espaços contínuos, fornecida pelo professor, sendo que se supõe que os parâmetros k_P, k_I e k_D excursionam no intervalo [0,+20] e que a planta é a seguinte:



Com $W(s) = 0$, $V(s) = 0$, $F(s) = 1$ e $C(s) = k_P + \frac{k_I}{s} + k_D s$.

Será realizada, portanto, uma busca empregando técnicas de inteligência computacional no espaço [0,+20] × [0,+20] × [0,+20], visando maximizar a função de *fitness*. Os critérios a serem atendidos são:

1. Tempo de acomodação menor ou igual a 0,5 segundos;
2. Tempo de subida menor ou igual a 0,04 segundos;
3. Margem de fase de 60 graus.

A função de *fitness* é tal que penaliza a violação das restrições na forma:

$$fitness(k_P, k_I, k_D) = \frac{1}{\left(p_1 + \frac{0.5}{0.04} p_2 + p_3 + 1\right)}.$$

tendo valor máximo em 1, quando todas as restrições forem atendidas. Cada termo de violação de restrição é dado por:

- $p_1 = 0$, se tempo de acomodação (t_s) é menor ou igual a 0,5 segundos e $p_1 = t_s – 0{,}5$ se tempo de acomodação é maior que 0,5 segundos.

- p_2 = 0, se tempo de subida (t_r) é menor ou igual a 0,04 segundos e $p_2 = t_r - 0{,}04$ se tempo de subida é maior que 0,04 segundos.
- $p_3 = (MF - 60)^2$, onde *MF* é a margem de fase do sistema de controle.

Se o aluno for trabalhar no Matlab, ele pode substituir o comando [**stepinfo_Q1**] pelo comando [**stepinfo**], *built-in* do próprio Matlab. Repare que, a cada proposta candidata de ganhos k_P, k_I e k_D, o programa implementa o sistema de controle e obtém a resposta ao degrau, permitindo assim calcular o valor de *fitness* associado. Também é usado o comando [**margin**] para se obter a margem de fase.

Existem soluções que atendem os três critérios de desempenho, levando a um valor de *fitness* igual a 1, mas não há garantia de se chegar a tal solução numa única execução do algoritmo. Para tanto, re-execute o algoritmo de busca até obter *fitness* = 1. Pode-se aumentar o tamanho da população (o valor *default* é 200) e/ou o número de gerações (o valor *default* é 50). Apresente os 4 gráficos obtidos após uma execução bem-sucedida e interprete cada um deles. Com 200 soluções candidatas por geração e 50 gerações, são avaliados ao final da execução 10.000 soluções candidatas.

Questão 2) Controle PID empregando PSO, que opera como uma meta-heurística de otimização em espaços contínuos.

Considerando o mesmo problema da Questão 1 e a mesma função-objetivo, proponha um algoritmo PSO (*particle swarm optimization*) para resolver o problema, encontrando os ganhos k_P, k_D e k_I. Para tanto, compare o desempenho das vizinhanças em anel e totalmente conectada. Apresente um pseudo-código de sua proposta de PSO e compare o desempenho com aquele produzido pelo algoritmo evolutivo da Questão 1. Como o problema tem um espaço de busca restrito, que deve ser respeitado, recorra, caso necessário, a estratégias abordadas na tese de Sabine Helwig ⟨https://opus4.kobv.de/opus4-fau/files/1328/DissertationHelwig.pdf⟩.

Questão 3) Otimização combinatória

Obter a solução de uma instância com pelo menos 50 cidades do problema de caixeiro viajante por mapa de Kohonen e por um algoritmo evolutivo. Nenhuma dessas propostas será competitiva com o estado da arte, por exemplo, dado pelo algoritmo Concorde (http://www.math.uwaterloo.ca/tsp/concorde/index.html), mas o objetivo é trabalhar os conceitos de auto-organização e evolução em computador, na solução de problemas combinatórios. No caso do mapa de Kohonen, devem ser apresentados resultados vinculados a múltiplas execuções do toolbox fornecido. No caso do algoritmo evolutivo, devem ser apresentados o pseudo-código, resultados intermediários, além do resultado final, e o comportamento da evolução ao longo das gerações (*fitness* do melhor indivíduo e da média da população).
Nota 1: Para o caso do mapa de Kohonen, é disponibilizado um toolbox, que pode ser utilizado como está ou então pode ser melhorado, de alguma forma. Para mais detalhes sobre a solução via mapa de Kohonen, consultar a tese no link ⟨ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/theses/tese_Lalinka_Gomes.pdf⟩, capítulo 4. O toolbox é de autoria do Prof. Levy Boccato (FEEC/Unicamp), que gentilmente cedeu o código e os demais arquivos de apoio.
Nota 2: No caso do algoritmo evolutivo, devem ser propostos múltiplos operadores de recombinação e de mutação, além de operadores de busca local.

Questão 4) Regressão simbólica usando Eureqa

Inicialmente, para um primeiro contato com o Eureqa, consulte material disponível em ⟨http://www.creativemachineslab.com/eureqa.html⟩. Em seguida, gere um mapeamento $\Re^1 \rightarrow \Re^1$, ou seja, $y = f(x)$, amostre dados para um intervalo restrito de entrada e empregue o software Eureqa, disponível livremente para uso acadêmico em ⟨http://www.nutonian.com/products/eureqa/⟩, para propor alternativas de funções de aproximação para a sua $f(\cdot)$. Faça o mesmo para um mapeamento $\Re^3 \rightarrow \Re^1$, ou seja, $y = f(x_1, x_2, x_3)$. Controle o intervalo de cada variável de entrada, produza mapeamentos desafiadores (mas evite muita composição de funções, ou seja, funções que são argumentos de outras funções), trabalhe com um bom número de amostras, insira ruído junto às amostras e evite uma composição de termos em que um termo predomine muito fortemente sobre os demais termos. Escolha adequadamente o conjunto de funções-base que serão utilizadas na busca (o usuário pode escolher as funções-base que irão participar da busca). Apresente a fronteira de Pareto obtida, com soluções de compromisso entre acurácia e simplicidade da solução, e escolha uma solução com boa relação de custo-benefício entre esses dois critérios (acurácia e simplicidade).

Questão 5) Redes bayesianas

Utilizando o pacote de software **Bayes Net Toolbox for Matlab**, em sua versão mais simplificada disponibilizada junto ao roteiro deste EFC2, siga o Manual do Usuário e construa uma rede bayesiana para o arquivo de dados fornecido no arquivo zip associado ao EFC2 (dados_BN_EFC2.mat) e, em seguida, faça:

(a) Apresente as características dos dados fornecidos: número de amostras, número de atributos, valores assumidos pelos atributos (no caso de atributos binários, 1 é falso e 2 é verdadeiro);

(b) Apresente a rede bayesiana resultante, incluindo as tabelas de probabilidades associadas a cada nó da rede. O acesso às tabelas de probabilidade deveria ser feito com o uso do mouse, ao clicar sobre o número da variável. Mas como passou a haver alguma incompatibilidade de recursos gráficos entre versões do Matlab (o toolbox recomendado corresponde a uma versão antiga e mais simples), a melhor opção é salvar a rede gerada (Network → Save), carregar o arquivo *.mat salvo (load .\network\filename.mat) e digitar [tables.cpt] na linha de comando do Matlab. As tabelas vão aparecer na ordem do número de cada variável da rede.

(c) Procure justificar por que o atributo 7 não possui conexões a outros nós.

(d) Qual é a probabilidade do atributo 5 ser verdade?

(e) Qual é a probabilidade do atributo 6 ser verdade dado que 5 é falso e 8 é verdade?

(f) Observando a tabela de probabilidades apresentada pelo atributo 1, qual é a relação lógica que você supõe existir entre ele e os atributos 8 e 9?

(g) Qual é a probabilidade do atributo 5 ser verdade, dado que 3 é verdade? Observação: Aqui são necessários cálculos a partir das tabelas de probabilidades, enquanto os itens (d) e (e) saem direto de campos dessas tabelas.

Observação: Para aqueles que não conseguirem ler o arquivo *.mat, há a opção em TXT, mas deve-se lembrar que o conteúdo do arquivo *.mat é uma única matriz denominada [data].

Questão 6) Árvore de decisão

Utilizando o pacote de software WEKA (http://www.cs.waikato.ac.nz/ml/weka/), construa uma árvore de decisão capaz de classificar os dados fornecidos no arquivo ⟨dados_AD_EFC2.arff⟩. Os passos são os seguintes:

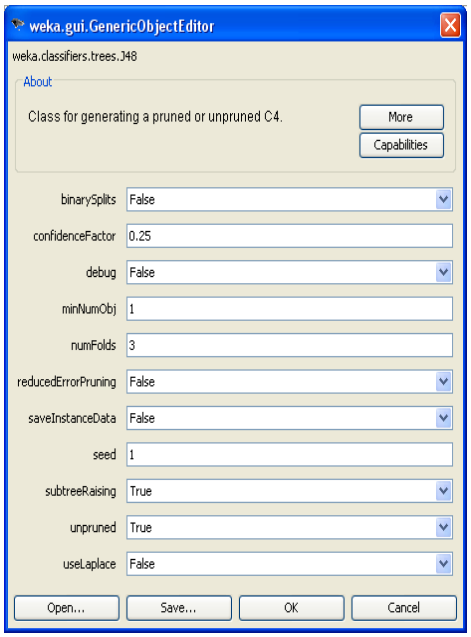
(1) Entre com a opção Weka Explorer.

(2) Na aba Preprocess, abra o arquivo fornecido (Open file). O arquivo ⟨dados_AD_EFC2.arff⟩ contém dados reais vinculados à presença ou não de um espécime em certos locais de observação, sob certas condições ambientais.

(3) Reporte quais são os atributos e estatísticas acerca de seus intervalos de excursão. Existem atributos numéricos? Existem atributos categóricos?

(4) Vá para a aba Classify.

(5) Em Choose, escolha trees/J48. Esta é a versão em Java do C4.5. Clique com o botão direito do mouse logo à frente de J48 e você vai abrir uma janela de parâmetros. Procure definir a configuração de parâmetros apresentada no quadro ao lado.



(6) Em Test options, escolha Cross validation Folds 10.

(7) Pressione o botão **Start** e analise os resultados. Indique a porcentagem de classificação correta e apresente a árvore de decisão. Apresente também a matriz de confusão, indicando o que ela informa.

(8) Apresente também a árvore de decisão na forma de regras SE ⟨?⟩ ENTÃO ⟨classe é ?⟩. Isso não envolve copy-and-paste.

(9) Altere ao menos os parâmetros **minNunObj** e **unpruned** (um de cada vez) e analise os resultados em comparação com o obtido na configuração anterior. Qual é o significado de **minNunObj**? Qual é o significado de **unpruned**?

(10) **Tome algum outro conjunto de dados de classificação, de preferência envolvendo atributos categóricos**, e ajuste os parâmetros do J48 visando bom desempenho da ferramenta. Repare que você vai precisar elaborar um cabeçalho nos moldes do arquivo <*.arff> fornecido. Analise os resultados, indicando a porcentagem de classificação correta e apresentando a árvore de decisão obtida.