O objetivo desta tarefa é desenvolver a capacidade de implementação de softwares a partir do paradgma de programação paralela. Para tanto, o discente deve desenvolver um programa paralelo incrementalmente usando as bibliotecas MPI ou OpenMP.

Sobre a aplicação a ser desenvolvida:

PROGRAMAÇÃO GENÉTICA LINEAR E ASSÍNCRONA

A programação genética é uma técnica de programação automática para a evolução de programas de computador que resolvem problemas. Começando com milhares de programas de computador criados aleatoriamente, uma população de programas evolui progressivamente ao longo de muitas gerações, utilizando, por exemplo, o princípio darwiniano da sobrevivência do mais apto.

A programação genética linear (PLG) é um método específico de programação genética em que os programas candidatos em uma população são representados como uma sequência de instruções baseadas em registradores de uma linguagem de programação imperativa ou linguagem de máquina. O adjetivo "linear" deriva do fato de que cada programa PLL é uma sequência de instruções e a sequência de instruções é normalmente executada sequencialmente.

Como outros métodos de Programação Genética, a Programação Genética Linear requer a entrada de dados para executar a população de programas. Em seguida, a saída do programa (seu comportamento) é avaliada em relação a algum comportamento alvo, utilizando uma função de aptidão. No entanto, a PGL é geralmente mais eficiente do que a programação genética em árvores devido os resultados intermediários (armazenados em registradores) poderem ser reutilizados e existe um algoritmo simples de remoção de íntrons que pode ser executado para remover todo o código ineficaz antes que os programas sejam executados nos dados pretendidos. Além disso, a PGL possui naturalmente múltiplas saídas, definindo múltiplos registradores de saída, e coopera facilmente com operações de fluxo de controle.

O sistema evolutivo assíncrono permite que os indivíduos evoluam de forma independente, ou seja, os indivíduos não precisam esperar pelas avaliações de outros indivíduos. A evolução assíncrona do programa tem o potencial de evoluir continuamente indivíduos (ou seja, programas) mesmo que os indivíduos não consigam completar sua avaliação, por exemplo, devido a um loop infinito.

COMPETIÇÃO E COOPERAÇÃO

O processo evolutivo é substancialmente baseado em competição. Todavia existem arquiteturas em que um sistema inteligente ou motor de resolução de problemas pode ser um componente separado que performa variavelmente dependendo da interação com estruturas de decisão representadas por genomas em evolução. O sistema inteligente funciona como um fenótipo ou abstração do processo biológico interno da bactéria. Uma bactéria pode conter qualquer tipo de sistema inteligente cujas restrições sintáticas possam ser implementadas por um programa que evolui separadamente. Todas as bactérias na mesma população devem compartilhar o mesmo tipo de sistema inteligente.

O PROJETO

ETAPA 1: CROMOSSOMOS E APTIDÃO

A primeira etapa desta atividade é projetar uma linguagem de programação binária com a quantidade mínima de 8 instruções diferentes, 1 operando por instrução, 4 registradores de E/S e programa de até 16 linhas de código.

O mecanismo de avaliação (fitness) deve ser assíncrono com capacidade de disponibilizar irrestritamente resultados parciais nos registradores E/S.

A entrega deve ser um código estruturado ou orientado a objetos contendo a estrutura de dados a ser utilizada como cromossomo, incluindo os atributos e seus significados para a programação, bem como o código que decodifica e executa cada instrução representada por partes do cromossomo. É desejável estruturas de código e dados compatíveis com aqueles disponibilizados no SIGAA.

ETAPA 2: VALIDAÇÃO DO CÓDIGO SEQUENCIAL DO GP
A segunda etapa mede a capacidade do sistema de evoluir uma população de códigos lineares síncronos com limite de tempo de execução
As tarefas são definidas na forma de benchmark

ETAPA 3: VALIDAÇÃO PARCIAL DO CÓDIGO PARALELO
A terceira etapa mede a capacidade do sistema de evoluir uma população de códigos lineares assíncronos com limite de tempo de execução seguindo uma topologia. As tarefas são definidas na forma de benchmark

ETAPA 4:
VALIDAÇÃO FINAL DO CÓDIGO PARALELO
Entrega do código final e relatório de atividades em PDF

ARQUITETURAS SUGERIDAS

1- MESTRE ESCRAVO
** MESTRE (GA) + ESCRAVOS (FITNESS EVALUATION)
** MESTRE (BACTÉRIA A EXPRESSÃO)) + ESCRAVOS (VARIABLES EVOLVED BY GA)

2 - SOFT DATA DIVISION
HOMOGENEOUS GA SUBPOPULATIONS.

3 - GRID GP
MULTITHREAD GRID VARIABLES V

BENCHMARK DE TREINO

F1: A+B-C-D = 0
F2: A+B > C=D
F3: A % B = 0
F4: A=B+1 && B=C+1 && C=D+1
F5: CH(A)+CH(B)+CH(C)+CH(D)+CH(A) = "OTIMO"

BENCHMARK DE TESTE: A SER DEFINIDO.

PRAZOS E REPOSIÇÃO - CALENDÁRIO DAS APRESENTAÇÕES

13/06/2025 Regular 1ª Avaliação - ETAPAS 1 e 2
27/06/2025 REGULAR 2ª Avaliação - ETAPAS 2 e 3 (II AVALIAÇÃO)
11/07/2025 REGULAR 3ª Avaliação - ETAPAS 3 e 4 (III AVALIAÇÃO)
25/07/2025 REGULAR 4ª Avaliação - ENTREGA DO RELATÓRIO EM PDF