Laboratório 3
Otimização com Métodos de Busca Local

CT-213
Prof. Marcos Ricardo Omena de Albuquerque Maximo

Carlos Matheus Barros da Silva
carlosmatheusbs@gmail.com

Março - 2019

Introdução

Foram estudados e implementados três métodos de otimização que possuem algoritmos baseados em busca local. Esses métodos foram Gradient Descent, Hill Climbing e Simulated Anneling.

Os métodos foram testados em um problema proposto no roteiro dessa atividade[]. Esse problema utilizou regressão linear para ober parâmetros físicos relativos ao movimento de uma bola.

Como o problema tratado possui solução analítica, o experimento é a titulo educacional, já que, dessa forma, pode-se obter facilmente sua solução pelo Método dos Mínimos Quadrados (MMQ).

De maneira geral, o Laboratório foi desenvolvido com sucesso. Em cada uma das sessões seguintes encontram-se o respectivo desenvolvimento de um dos métodos estudado.

Gradient Descent

Enunciado 1. *Faca um programa em que três processos executam paralelamente as seguintes ações:*

- *Pai - Imprime os números de 1 a 50, com um intervalo de 2 segundos entre cada numero. Apos imprimir todos os números, imprime a frase "Processo pai vai morrer".*
- *Filho1 - Imprime os números de 100 a 200, com um intervalo de 1 segundo entre cada numero. Antes de imprimir os números, imprime a frase "Filho 1 foi criado", e após imprimir todos os números, imprime a frase "Filho 1 vai morrer".*
- *Neto1- (filho do processo Filho1) imprime os números de 300 a 350, com um intervalo de 2 segundos entre cada numero. Antes de imprimir os números, imprime a frase "Neto 1 foi criado"Apos imprimir todos os números, imprime a frase "Neto 1 vai morrer".*

Em cada printf os processos devem imprimir o seu pid e o pid do seu pai.

Mostre as saídas enfatizando os processos executando em "paralelo".

Resposta 1. O Problema 1 foi resolvido com sucesso. Seu código pode ser conferido no Código **??**. Seu funcionamento pode ser observado pela saída representada pelo Código **??**. As respostas às perguntas referentes a esse problema podem ser vistas na Discussão 1.

Discussão 1. O resultado foi obtido dessa forma devido ao fato de que a uma das primeiras coisas que o programa pai faz é chamar *fork()* de modo que logo em seguida seu filho chama *fork()* também. Então ficaram os 3 processos rodando paralelamente, de modo que cada um desses 3 começaram a fazer o que estava descrito no enunciado do problema.

Dessa forma, como pôde ser visto na saída do problema, os três processos estavam executando paralelamente. Como cada um desses 3 processos *printava* "simultaneamente" foi visto que o Neto foi *printado* ante do filho, o que é algo possível, já que como todos estão rodando mais ou menos na mesma posição o que chamar a função *printf* primeiro seria *printado* antes.

Os três processos ficaram em seus ciclos *printando* de acordo com o previsto, até que eles morreram. O pai, como era previsto, morreu primeiro, já que ele havia começado a *printar* antes. Logo em seguida, o filho e o neto também terminaram.

Problema 2

Enunciado 2. *Altere o programa 1 para que primeiro sejam exibidos primeiramente apenas os prints do neto, depois so os do filho e por ultimo so os do pai.*

Resposta 2. O Problema 2 foi resolvido com sucesso. Seu código pode ser conferido no Código **??**. Seu funcionamento pode ser observado pela saída representada pelo Código **??**. As respostas às perguntas referentes a esse problema podem ser vistas na Discussão 2.

Discussão 2. A alteração necessária para a mudança requisitada foi bem simples e direta, já que o que foi pedido foi que rodasse o neto antes do filho, bastava fazer o filho esperar terminar a execução do neto para então executar o que ele ia fazer, e como foi pedido também que o filho rodasse antes do pai, bastava que o pai, também, esperasse o termino da execução do filho para que então executasse o que ia executar.

Portanto foi necessário apenas acrescentar duas linhas com a função *wait()*, a Linha 34 e a Linha 40.